UNIVERSITY OF OSLO
Department of informatics

Internationalization and localization
A case study from HISP

Master thesis
60 credits

Øyvind Fladberg Brucker

01.08.2007

Abstract

Translation of computer software is commonly separated into two key terms, internationalization (i18n) as an enabling factor and localization (l10n) as the process of translating the software to be suitable for a specific context. With an emphasis on distributed development and software design, I present the internationalization and localization efforts of a Health Information System (HIS) situated in the context of developing countries.

Through an action research approach, which includes a 4 month field study in Vietnam, I investigate the use of open source software (OSS) in developing countries. In collaboration with a Vietnamese team, I developed the first internationalization solution for the second generation of the District Health Information Software (DHIS 2), and localized the software for Ho Chi Minh City (HCMC). DHIS 2 is an OSS project and a product of the Health Information Systems Programme (HISP), which is an organization dedicated to improving health care in developing countries. Based on my discoveries, I discuss the establishment of localization teams in developing countries and their integration into a global OSS project.

I point to a shift of focus on internationalization and localization as processes of isolating culture specific data, to a broader perspective on how organizational structures and technology can be adapted to support such change. Modularization and how general solutions can be reached are discussed based on cases from the development of DHIS 2, and a digital divide between developing nations and OSS communities in the effective use of communication technology is identified.

Internationalization is traditionally situated at the presentation layer of applications. Based on an implementation in the DHIS 2 project, I present an aspect oriented concept for enabling internationalization on the lower layers of software projects, and how this concept can be aligned with existing installations.

Acknowledgements

I would like to thank Knut Staring for his valuable feedback and guidance in writing this thesis, and the respondents of my survey for taking time out of their busy days to respond to all of my questions.

I would like to thank my family, my Vietnamese friends for their hospitality during my field study, the DHIS 2 developers who devote a remarkable amount of time into making the second generation of DHIS possible, the HISP coordinators for their guidance and support and Jørn Braa for convincing me to travel to Vietnam.

Table of Contents

1. Introduction

This thesis explores the processes of internationalization and localization in an open source software (OSS) project. The need for localization is most urgent in developing countries, as most software is written for and by western countries (Esselink, 2000), hence my research focuses on developing countries that currently face these challenges. The scope of this thesis covers both the enabling factor of internationalization and the process of translating software to a specific locale (localization). The process of enabling internationalization functionality in existing systems is generally regarded as a complex task that is situated at the higher layer of applications (Kersten et al., 2002). I discuss how internationalization capabilities can be enabled at both the higher and the lower layers of existing systems.

My empirical base is my participation in the OSS project District Health Information Software version 2 (DHIS 2), which is a product of the Health Information Systems Programme (HISP). I have participated in the project since the spring of 2005, which includes a 4 month field trip to Vietnam.

1.1 The action research project

HISP was founded after the fall of Apartheid in South-Africa in 1994, and it has since then had a goal of improving health services in developing countries (Braa and Hedberg, 2002). It has since its conception focused on participatory design through action research, where all actors are included in the development process. The tool it uses to reach its goal is the District Health Information Software (DHIS), which is installed at health facilities and used to register data. The choices of technology of the first generation of DHIS were taken mainly based on what was available locally in South-Africa, and what the actors involved had competence in (ibid.). As the project expanded to other countries, difficulties with both the technology and the organizational structure in which it was developed was starting to show (Nordal, 2006). This lead HISP to start investigating the possibility of creating a new generation of DHIS (DHIS 2) which would be based on open source technology only and that could be organized as a global development

project. Participation is an essential part of the action research approach of the HISP project, where researchers in addition to observing the situation are directly participating in it.

DHIS 2 is OSS and free both in the sense that HISP do not charge anything for the use of it, and users adapting it are free to modify and redistribute it. HISP establishes contact with governments and participates in the localization and implementation of DHIS. I have been part of such a process in Ho Chi Minh City (HCMC), Vietnam, and participated in the development of the DHIS 2 application over two years. I was introduced to the action research project through a course at the University of Oslo (UiO) in the spring semester of 2005. This course serves as an introduction and a base for recruitment of master students to the DHIS 2 project, and students are after an introduction to the technologies actively participating in the development of DHIS 2.

1.2 Motivation

OSS is software with a license that allows anyone to modify and redistribute the source code (Weber, 2003). The literature on OSS internationalization currently have a technical angle, and the term internationalization, or its abbreviation “i18n”, generally refer to the task of translating text, formatting numbers and dates and specific technical challenges (Kubota, 2006). While including technical aspects and with a focus on OSS, I maintain a broad perspective on how internationalization solutions can be established and used to enable localization. I look at how this relates to the phenomenon referred to as the digital divide, which concerns a divide in the lack and the effective use of information technology globally (Gurstein, 2003). The HISP case have a combination of characteristics which from my perspective makes it unique, as it is based on new open source technology and it faces some very demanding requirements from local authorities in the health sector of developing countries. I investigate how the flexibility to meet such requirements can be kept, while keeping a standardized product globally. While there has been a lot of study into the translation of software (localization), I also focus on enabling alternatives such as generality in software development processes and its relation to internationalization.

I chose to work in the DHIS 2 project prior to the field of internationalization, and the decision to focus on this was also reached based on what was needed in the project at the time. The main focus of this study is on developing countries, which can be attributed to two main reasons. It is the main concern of the DHIS 2 project, and most localization efforts are done from western to developing countries and emerging economies (Esselink, 2000). The focus on open source development in developing countries presents challenges beyond the perspective of open source literature, such as Fogel (2005), who mainly focuses on how to get and keep the attention of developers already familiar with the communication frameworks. The HISP project is established in countries with cultures that are unfamiliar with the concept of OSS (Weber, 2003), which makes it an interesting case for the study of global development socially.

Based on a case from India where DHIS 2 is implemented, I got the opportunity to explore how internationalization functionality can be aligned with the lower layers in software projects. At the time this case was presented, DHIS 2 was installed at several locations in both India and Vietnam. Hence, to keep a standardized application globally it was necessary to align the new solution with the existing installed bases. This case introduced challenges beyond the traditional view on internationalization and software design (Kersten et al., 2002). While such solutions can be aligned with existing software projects and database systems by doing changes to the data model (Sun, 2007a), the case introduced through HISP fostered a more generic concept because of a request to limit such changes.

Personally, my motivation for working in the DHIS 2 project was initially based on the technical challenges it presented. The global aspect of the project was not very apparent to me in the spring of 2005 as only Norwegians did the development. Although one central Norwegian developer was situated in Vietnam working with Vietnamese developers, only the Norwegian developer was visible through our online communication. The system was also in its early conception phases and far from being ready for a public release, all in all making this more of a local development effort than a

global OSS project. This perspective changed somewhat gradually when the first milestone of DHIS 2 was released and development nodes in developing countries started participating in the online discussions. Local teams were hired both in Vietnam and India, contributing to the evolving global network. When I got the opportunity to travel to Vietnam on a field trip, social aspects also caught my interest. The opportunity to develop the first part of the internationalization solution of DHIS 2 along with local developers presented itself a great opportunity for me personally.

The focus on OSS along with the overall goal of the HISP project of contributing to better health care in developing countries is something that I find very interesting. I also find the area of internationalization and localization fascinating, particularly because it is at such an early stage. I regard localization as something that is likely to be a part of almost any global software product in the future, especially because of the growing amount of services available directly over the internet. DHIS 2 is in this regard a very interesting product to investigate. Not only because it's open source, but also because of the contacts it has through the action research project which allows us to develop and test the solution with real users globally. It also allowed me to gain experience from a real global software project, which provides a perspective on development outside of a closed research community at the university. Additionally, with a background from computer networking research I have also been a frequent user of OSS, which made me interested gaining more incite into how open source projects and its culture.

1.3 Research objectives

In this thesis I will focus on internationalization and localization as processes. A significant part of this research was carried out in the context of a localization effort in Ho Chi Minh City (HCMC). My main focus will be on the software as a global OSS product, in the context of a Health Information System (HIS). The medical reporting standards were not changed through my participation in the project, but these will be discussed in relation to localization process in which they were reimplemented. I will focus on both internationalization as the enabling factor and localization as the process of

adapting a system to a specific location and culture. This led me to the following research objectives:

Primary research objective: *Explore challenges in internationalization and localization of open source software in the context of developing countries, with an emphasis on:*

- *The development of internationalization solutions and the establishment of supporting infrastructures.*
- *Local and global tensions in relation to software design.*
- *The establishment and integration of localization teams into global development networks.*

Based on a requirement that emerged from the health sector in India and the DHIS 2 application, I will in my secondary research objective develop a concept for the enabling of internationalization of the data model of a software project.

Secondary research objective: *Explore how internationalization can be enabled on the persistence layer in software development projects with an installed base.*

1.4 Structure

Chapter 2: Literature review

Presents relevant theoretical backgrounds for my thesis, which will be discussed in relation to my empirical findings.

Chapter 3: Methods

Presents the methods used in my study, and describes how they have been applied in the context of my research. Action research is presented and related to the field of information systems, and method related issues are identified. The survey research method is presented, along with a description of my approach to the application of the methods.

Chapter 4: Case

The HISP project is presented along with the two generations of DHIS, with an emphasis on DHIS 2 and its communication practices and technology. Additionally, I provide background information about the Vietnamese context in which I carried out my field study and I present an internationalization case that emerged from India.

Chapter 5: Empirical study

My empirical study consists of my findings during my work in the HISP project from the spring of 2005 through the spring of 2007, with emphasis on a 4 month field study in Vietnam. Lastly, the result from a survey of the development nodes in the DHIS 2 project is presented.

Chapter 6: Discussion

This chapter contains a discussion of my empirical findings.

Chapter 7: Conclusion

This chapter concludes my research.

2. Literature review

I will in this section present relevant theoretical background for my thesis, which will be discussed in relation to my empirical findings.

2.1 Information Systems as social systems

In the 1970s and 1980s, organizations were the major sites of computerization, and hence the information systems (IS) research (Kling et al., 2000). At that time computer software was seen merely as tools to complete specific tasks, and the research was generally deterministically phrased (ibid.). An increasing level of computerization in private homes, along with a vast growth in public access to the internet (Wikipedia, 2007e), has changed the perspective of IS research.

Kling et al. (2000) advocate the use of *social informatics* as a common term that merge IS related sciences which take the interaction of information systems with institutional and cultural contexts into account. An array of relevant factors is related to social informatics including social, cultural, organizational and other contextual components. Kling et al. (2000) argue that work processes and practices need to be studied for how they are carried out, and that information and communication technology (ICT) is more usefully conceived of as sociotechnical networks than as tools.

2.2 Culture in information systems

Studies of information systems and internationalization reveal that all information systems are to some degree socially shaped. Technology is more than just equipment as it also incorporates a surrounding shell of infrastructural requirements, technical and managerial skills that are needed in order to operate it (Baark and Heeks, 1998). Management, communication and collaboration depend on culture and evolve together with social systems, and all information systems capture some degree of the culture in which they were created (Kersten et al., 2002). Hence, there are no unifying scientific principles that can abstract away the fact that people use software in an extraordinarily diverse technological and cultural matrix that changes almost continuously (Weber,

2003). The big majority of software is developed in the western world (Esselink, 2000) leading information systems to include particular social and cultural assumptions that may not apply in developing countries (Baark and Heeks, 1998). Kersten et al (2002) present two opposing perspectives on culture:

Holistic perspective: There is neither a universal culture nor universal laws, hence the set of symbols unique to a given culture cannot be detached and interpreted as an instance of a culture-free biological basis.

Reductionist perspective: Culture as a symbolic discourse and a language with universal laws in which the creation of sets, schemas and networks for symbol manipulation can be created.

These cultural perspectives are further related to three key technological perspectives (Kersten et al., 2002):

Instrumental perspective: System design and development can be done in isolation of the users and their situation. Technology proceeds separately from cultures, values and societies, resulting in an uncritically positive and somewhat self-limiting view of technology.

Substantive perspective: System design and development can be for the betterment of the users leading to a new social and cultural values and systems. Information systems can socially and culturally both have positive and negative effects.

Critical perspective: System design and development is never neutral and can be used to propagate and infirm social and cultural values and systems. Hence, technology can be used to advance and enrich social objectives, and technology and information systems cannot be seen as separate from people.

The emergence of commercial globalization has lead to a pragmatic view of culture in relation to information systems. This has caused a reductionist understanding of culture and an instrumental technological implementation (Kersten et al., 2002). Kersten et al. (2002) further advocate a more holistic perspective, using a substantive or critical perspective on technology which suggests that software technologies are not culturally neutral. The scope of a master thesis requires me to keep a quite pragmatic view of internationalization from a technical point of view. I will however maintain a holistic perspective on technology, and through the use of action research have both a critical and a substantive standpoint.

2.3 Digital divide

The digital divide refers to a growing technological gap between the educated and the uneducated, between economic classes, globally and between the more and less industrially developed nations (Gurstein, 2003). Gurstein (2003) argues that there should be more emphasis on the effective use of the technology, and that this should be the main concern of the digital divide in contrast to just availability. He goes far in suggesting that the understanding of the term digital divide in its current form is no more than a sales pitch from providers of communication technology, and argues for a deeper look at the Internet as a fundamental tool in a new digital economy (ibid.). Digital divide also applies to the level of technical skill and computing power available, which usually are prerequisites for effective use of communication technology (ibid.).

2.4 Technology transfer

Based on a case study with focus on transfer from western to developing countries of four individual projects, Baark and Heeks (1998) presents framework for evaluation of technology transfers. The outcomes of technology transfer are discussed, with an emphasis on the adaptation of technologies to local conditions and the sustainability of capabilities created. At the core of the framework is the continuous *information technology transfer life cycle:*

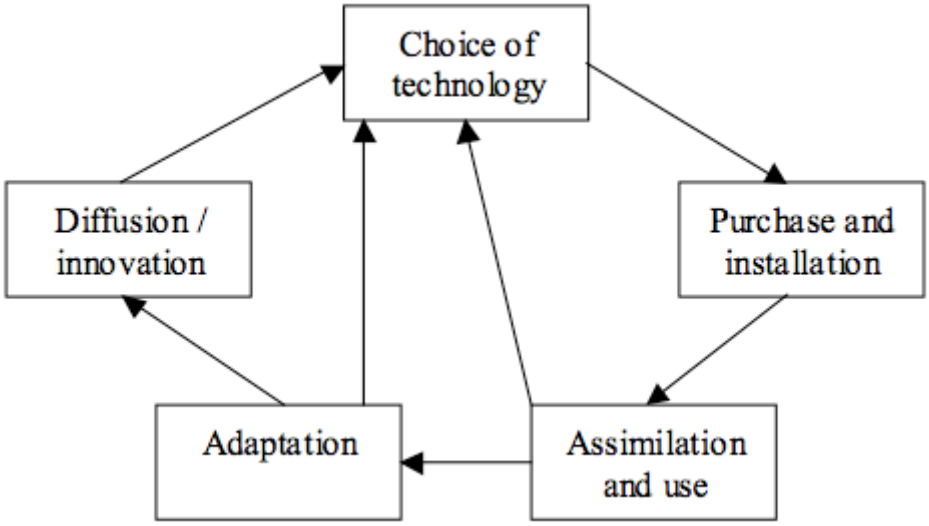

Figure 1: The information technology transfer life-cycle from Baark and Heeks.

This is presented as a logical and practical, rather than theoretical framework for technology transfer. Starting from the top, choice of technology is the phase where alternatives are surveyed and decisions on core technology are taken. The following stage includes the actual purchase of the technology and basic training in its use. During assimilation and use the people that work with the technology are further trained in the use of the technology and its regular maintenance. The next stage of adaptation includes localization through modification of the technology, as indicated by the figure this stage is not always present. Adaptation may lead to innovation and diffusion (Baark and Heeks, 1998). This thesis touches upon all the phases in the technology transfer life-cycle, with an emphasis on assimilation and adaptation.

2.5 Global software development

Globalization is increasing the need for software companies to distribute their development, and large scale investments on infrastructure are no longer a terminally limiting factor in whether or not organizations can undertake global work (Sahay, 2003). The ways organizational environments are being reshaped for this new scenario have been given various labels, such as the “new economy”, “digital economy”, “network society” and “information age” (ibid.). Global development can conceptualized in the term Global Software Work (GSW), which is defined as:

"Software work undertaken at geographically separated locations across national boundaries in a coordinated fashion involving real time and asynchronous interaction." (Sahay, 2003, p. 1)

The literature indicate that the distribution of tasks implies a significant cost, Herbsleb and Mockus (2003) suggests that a factor of 2.5 for the time it takes to complete tasks from same-site development to a distributed environment. This factor is largely based on size, diffusion and number of people involved, and it's further suggested that more people are required in order to perform tasks in a distributed environment than in the case of same-site work (ibid.). Interdependence between sites are identified as a contributor that slows down global development, and decoupling of work so that sites can work independently of each other is strongly recommended (ibid.). Several important criteria's for successful communication and coordination seems to be lost or severely reduced in a distributed environment, such as information about expertise, context, status, workloads, urgency and availability (ibid.). Herbsleb and Mockus (2003) suggests that the loss of informal, or "water cooler", communication channels may be a primary reason for this loss of information, and suggests the use of text based communication such as instant messaging to restore this communication channel.

2.6 Local and global tensions

The term localization implies that some modification of the software is necessary in order to make it applicable in local contexts. Such local requirements unveil a dilemma for global development between the benefits of having a global standardized product, and the flexibility to customize the product locally. Joshi et al. (2006) advocate a balanced consideration of interests and interpretations throughout the hierarchy of organizations, in order to tame local and global tensions:

"The design of global information systems needs a persistent focus on the work practices at local level in order to achieve the intended support for the work, and in order to ensure that the system at least does not contradict the employees' own work practices. In

addition, paying attention to the minor but locally relevant features of the system can go a long way in assisting the diffusion of such collaborative technologies within a global organization." (Joshi et al., 2006, p. 29-30)

Joshi et al. (2006) point to how users of a system find "workarounds" and are reluctant to use system if they conflict with existing practices, and emphasizes the need of organizations to acknowledge and embrace such diversity instead of directly imposing one-sided uniform directives.

Rolland and Monteiro (2002) explore the tensions between the local and global in the context of a survey system for ships serving 300 ports in 100 countries worldwide. With a focus on how to strike a pragmatic balance between the sensitiveness to local contexts and the need to standardize across context, they look at what "costs" actors need to pay in order to achieve working solutions (ibid.). Pollock et al. (2007) argue that it exist a narrative bias among researcher on how cultural differences are treated in information systems, and advocates more reflection about the relation between computers as *singular* and *monolithic* systems and how the translation of software are bridging the gap between the standard and unique when brought together. Further, they argue that information systems can benefit from focusing on similarities *across* cultures, rather than the primary focus on development western cultures and then export by isolation of culture specific aspects and localization (ibid.).

2.7 Internationalization

In Information systems research internationalization is commonly regarded as a factor that enables translation of computer software into local contexts. It is difficult to create and nearly impossible to maintain isolated "democratic islands" of localized information systems (Braa and Hedberg, 2002). Hence, it is necessary for computer software to support local contexts while maintaining a standardized product globally. This can be analyzed in the terms of *standardization* and *flexibility* for localization (ibid.). While standards are foundational for coordinating activities across time and space, flexibility is necessary for successful localization (ibid.). As defined by Nhampossa and Nielsen

(2004), internationalization builds on standards to enable localization. Internationalization is the process of designing a product so that it can be easily localized without the need for redesign. In other words, it is the process of designing and implementing a product which is as culturally and technically "neutral" as possible, and which can therefore easily be localized for a specific culture or cultures (ibid.). Nhampossa (2004) identifies two types of applications in relation to internationalization:

General Business Domain Application (GBDA): Refers to general purpose software, such as text processors and spread sheets. In this, the functionality, content and the interface is largely decided by the software vendor, and it is relatively easily used across organizations, countries, contexts and cultures.

Special Business Domain Application (SBDA): Software of this type is more focused on application, and the translation process requires a greater understanding of the business domain and the context of use. This often requires a participatory effort where the software developer is taken directly into the context of implementation with the end-user.

The translation process of SBDA software is hence regarded as more complex and costly compared to GBDA applications. In this thesis I focus on the translation of OSS SBDA in the context of developing countries. Internationalization is understood as developing software systems to support localization for specific locales, for example to its different language, standards, legal requirements and cultural norms (Nielsen and Nhampossa, 2005). Hence internationalization often takes place prior to localization during development, but it should not be regarded as an isolated process:

"Internationalization and localization should neither be understood nor treated as different and subsequent processes. Such a distinction does not take into account the interrelatedness of internationalization processes and can easily mislead us to interpret internationalization success or failure as solely determined by centralized efforts of internationalization and control." (Nielsen and Nhampossa, 2005, p. 9)

Internationalization reduces the time and resources required for product localization, thus saving producers money and improving their time-to-market with local version (LISA, 2007). Internationalization is commonly used interchangeably with globalization (g11n) to refer to economic and cultural effects of an increasingly interconnected world (Wikipedia, 2007d). LISA (2007) defines globalization as:

"Globalization addresses the business issues associated with taking a product global. In the globalization of high-tech products this involves integrating localization throughout a company, after proper internationalization and product design, as well as marketing, sales and support in the world market." (LISA, 2007)

Globalization is hence defined in the commercial market as the process of positioning a company in a global perspective, and internationalization is an enabling factor for such a process. I will in this thesis relate internationalization to software design and OSS development in developing nations.

2.7.1 Scope

The term internationalization is used for anything from being synonymous with globalization to just the technical process of enabling translations of the user interface in computer software. As defined by LISA (2007) internationalization is a process that enables localization by writing software neutrally. Internationalization will in this thesis refer to technical solutions that enable translations of software for different cultures, but I will include cultural and social considerations beyond language translations. Many illustrative aspect such as colors, icons and images are only meaningful within restricted cultural contexts, one can not just assume that these will be universally understood (Dix et al., 2004). Hence, the field of internationalization and human-computer interaction (HCi), which focuses on all aspects of the interaction between humans and computers, is closely related with somewhat overlapping scopes.

There is little doubt that extended cultural considerations will increase the usability of software (IBM, 2007), but the problem lies in the development cost of such customizable

solutions. Based on a case study in Rwanda, Reinecke and Bernstein (2006) reveals how cultural considerations can have significant value for end users, but they also point to how tedious this process is. The problem is twofold between technical development and gaining sufficient cultural knowledge. They conclude that such processes can't justify the cost of development, and point to implementation of artificial intelligence to enable flexible user interfaces (ibid.). The process of enabling extended cultural considerations, or *culturability*, is by many regarded as the next big leap in internationalization technology (ibid.). The scope of a master thesis requires me to have quite a pragmatic view of internationalization in regard to its technical scope, but I will maintain a broad definition of the term.

In relation to software design, the generic approach often required by internationalization is in many ways very similar to the enabling of accessibility. This includes enabling a system for people with sensory (hearing and vision), motor (orthopedic) and cognitive (learning, speech, mental) disabilities. This is generally not regarded as a part of internationalization, and not within the scope of this thesis.

2.7.2 Motivation

"Poorer countries would also benefit from technology which helped developers to produce systems for less substantial markets. The benefits to organizations are obvious: an opportunity to boost international market share, and simultaneously reduce risk by diversifying their served market." (Mahemoff and Johnston, 1998)

Software publishers are by the use of internationalization able to ship as much as thirty localized versions of their software at the same time as the original, this is often referred "sim-ship" (short for "simultaneous shipment") (LISA, 2007). There are many obvious arguments that motivate towards enabling i18n for a solution such as bigger target audience by increased availability. Mahenmoff and Lorraine (1998) points out how synergies can be gained by using the same system as others to draw nations closer together, thus contributing to globalization. In the case of HIS, internationalization can lead to further adoption of international standards, such as the Millennium development

goals set by the World Health Organization (WHO) (WHO, 2007b) and the use of ICD codes. ICD is an acronym for International Classification of Diseases, and it includes standardized world-wide comparable codes for diagnostic classification and many health management purposes (WHO, 2007a).

2.7.3 Technical implementation

The common way of implementing internationalization in information systems is to separate the information for each location and the most common use is to multi-language enable applications (Esselink, 2000). The process of translation in this context is often used interchangeably with the term language localization (Wikipedia, 2007f). For language translations this process commonly implies to define a common identifier for each whole sentence or paragraph, and create localized versions of that identifier.

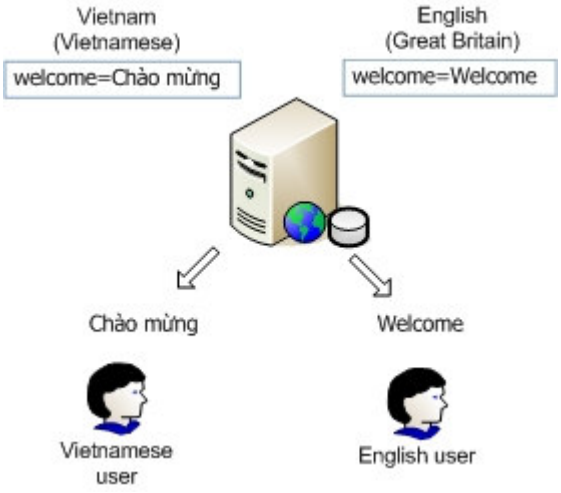

Figure 2: Internationalization keys

This solution is very static, but the complexity and diversity of languages makes this the only conceivable solution. The common identifier is often referred to as the "i18n key". The abbreviation i18n is mainly used in technical forums possibly reflecting the laziness of the average programmer; it's simply a compression of the word internationalization where the 18 refers to the amount of characters between the "i" and the "n" (Wikipedia, 2007d). The term translation is also used in a broader sense of translating the software and not only the language in the software, though language translation is the most common use (Esselink, 2000). This often leads the terms "i18n" and internationalization to be used from a technical perspective as synonymous with enabling language

translations only. Internationalization will be further discussed in relation to software development in section 2.11. In April 2007 the World Wide Web Consortium (W3C) released a new standard for defining internationalization of XML content named Internationalization Tag Set (ITS). This is a standard that standardizes many aspects that have been difficult to handle consistently up until now, such as inclusion of quotes in different languages and consistent escaping of translations (W3C, 2007).

Creating a general translation framework between vast amounts of languages supporting interconnected translations are not a task that's currently possible. Various *machine translation* technologies such as Babelfish (AltaVista, 2007) do exist, and a big collection of them are freely available (I18nGurus, 2007).While these are valuable for tasks like understanding key points of texts in an unknown language, they are not considered safe to use for general purpose translations. Even if it was technically possible to incorporate the grammar of the languages for automatic translations, there are other complexities relating to cultural differences (Dix et al., 2004) that may cause such a process to yield incorrect results.

2.8 Localization

Localization is the adaptation of a product to make it appropriate for a target region. While internationalization is the enabling factor, localization is the actual process of customizing a solution. In the case of software this often includes modifying screen dialog boxes, field lengths, date, time and currency formats, icons and performing language translations. Mainstream business applications such as address databases and financial accounting packages often have to be adapted to the procedures and conventions applicable in their new environments (LISA, 2007). Because the majority of software is being developed in English, approximately 80% of the localization effort is performed with English as its base language (Esselink, 2000). Input systems are programs that provide the ability to enter complex script languages using standard keyboard configurations, and to establish such solutions can be an essential part of localization processes (Wikibooks.org, 2007).

2.8.1 Scope

The scope of localization goes beyond factors enabled by internationalization, and it can as a term be applied to other products than just software. Translations of a book or adding subtitles to a movie can be regarded as localization of those products. Such translations often include more than just word by word translations. Aspects such as date, time and currency will often also need to be converted to the local context. In static text this can be typed in directly, but in computer programming a more dynamic approach is needed. The de facto standard method of handling this is to define *patterns* for this during the localization processes, which is later processed by computer based on a *neutral* format. In relation to the computer industry the term localization can be applied to translation of user documentation and more complex topics such as translations of legal documents. Localization requirements can often be dictated by social perceptions as well as laws and regulations, such as the inability of a car manufacturer to sell a car with the steering wheel on the left side in the UK or to sell it without providing a user manual containing safety instructions in a local language (Esselink, 2000). Similarly to i18n (ref. 2.7.3), l10n is often used as an abbreviation for localization (ibid.) Yeo (1996) distinguishes between *overt* and *covert* factors in localization processes:

Overt: Includes the commonly known facts about a culture, such as time factors, writing direction, special characters, word ordering, use of jargon, units for measurements and date/time formatting.

Covert: Includes factors that are more complex to uncover related to cultural preferences. This includes mental dispositions, perception, social interaction rules and context of use.

2.8.2 Commercial localization

The need for localization has spawned a new market of specialized localization companies, which over the past decades has developed into being a multibillion dollar industry (Esselink, 2000). The Globalization and Localization Association (GALA) is a non-profit organization created to promote this industry. It was founded in 2002 by 15

localization companies from 12 countries on four different continents, and their database does now include thousands of companies (GALA, 2007). In an effort to give the software a feel of consistency among cultures, commercial companies commonly provide glossaries that serve as guidelines for translators, and run their own verification tests in house before releases (Esselink, 2000). Localization of OSS projects will be described in more detail in section 2.10.6.

2.9 Standards

There are two standards that are essential to internationalization and localization and they are both maintained by the International Organization of Standardization (ISO) (ISO, 2007c). In addition to these there are commonly available a less restricted property often referred to as *variant*. A combination of definitions from these standards is referred to as a *locale.*

2.9.1 Country identifier (ISO-3166)

ISO-3166 as defined in ISO (2007a) defines country codes. Several versions exist of this standard. The most commonly used version is the ISO-3166-alpha2, which contains two character codes only. This is both the standard in the Java programming language used in the development of DHIS 2, and the top level domain names (TLDs) of the Internet. This standard also contains a few assigned codes that are used as TLDs, but doesn't refer to a country directly such as "uk" and "ac" (ISO, 2007b). The term "country code" used throughout this thesis refers to ISO-3166 codes.

2.9.2 Language identifier (ISO-639)

ISO-639 as defined in ISO (2007d) defines the languages and will obviously have requirements vastly exceeding the two character space limitation of ISO-3166. Several versions of this standard is developed or are currently in development ranging from a character space of two in alpha-2 (ISO 639-1) to a character space of 4 in alpha-4 (ISO 639-6). The 3 character alpha-3 (ISO 639-2 and ISO 639-3) has a much wider scope than ISO 639-1. It also includes identifiers for language collections and it's less restrictive than ISO 639-1 (LibraryofCongress, 2007).The term "language code" used throughout this thesis refers to ISO-639 codes.

2.9.3 Variant

Variants are used to add an additional property to the country and language identifiers. I have not been able to obtain any official standard for this property, which has many different usages and very little restrictions in terms of character space. Sun Microsystems (2006) describe it as “vendor or browser specific” and exemplifies it to identify information such as platform, browser type and dialects.

2.9.4 Locale

The term locale generally refers to the realization of the cultural variable in software, such as “Japanese”, or “French-Canadian”. Since culture goes beyond geographic details, it follows that locales can do likewise (Mahemoff and Johnston, 1998). In the Java programming language locale definition (Sun, 2006) only the language code is required, but the addition of country adds precision to the definition. For instance French is used both in France and Canada, but precise usage and idiomatic expressions vary between the two cultures (O'Conner, 2005). When I refer to the term *culture* in this thesis, a locale can be used to identify that particular culture technically. The term localization was derived from locale, but locale is generally now used in a more technical contexts (Esselink, 2000).

XenCraft (2005) argues that the current locale standard is failing by pointing to a cost in terms of interoperability and standardization across contexts, and further argues how the locale fails to capture the users cultural preference. There is also a lack of standardization of the definitions for parsing of locale hierarchies (XenCraft, 2005):

- RFC 1766, XML doesn’t provide guidance.
- HTML 4 follows locale hierarchy –fr-FR-euro, fr-FR, fr.
- Java follows locale hierarchy, then default locale hierarchy.
- Unix seeks locale, then defaults to “C” locale.

2.10 Open source

The term OSS describes software with a license which allows anyone to modify and redistribute the source code (Edwards, 2001). Open source is used as a term to denote a

set of principles that promotes information sharing. Licenses based on open source principles can also be applied to other areas than software such as literature (Raymond, 2000), but OSS will be the main focus of this thesis.

Open source refers to freedom and not free as in a free product or service. A common confusion around the word “free” is related to an ambiguity in the English language, which lacks the distinction between low prices and liberty. In Roman languages this distinction is immediately clear by the terms gratis and libre (Fogel, 2005). Hence, the term “free software” in the context of OSS refers to the freedom to modify and redistribute the program, and not only the fact that it is free of charge. To clarify the confusion caused by the this language ambiguity, the term “open source” was established by the Open Source Initiative (OSI) in 1998 (OSI, 2007). It also serves as an initiative to make the case of open source to the commercial world:

“The Open Source Initiative is a marketing program for free software. It's a pitch for "free software" because it works, not because it's the only right thing to do. We're selling freedom on its merits. We realise that many organisations adopt software for technical or financial reasons rather than for its freedom. Many users learn to appreciate freedom through their own experience, rather than by being told about it.” (OSI, 2007)

The Free Software Foundation (FSF) was established in 1985 by Richard Stallman and it’s closely tied to the GNU project (Fogel, 2005). The GNU project creates OSS and promotes information sharing. While the OSI is aimed at the commercial world, the FSF has more of an ideological base. FSF believe that all information should be free, whereas OSI will let the users decide for themselves whether it should bee free or not (ibid.).

2.10.1 Software

OSS has emerged from academic and research communities and been widely adopted in the market of internet services (Kogut and Metiu, 2001). The background for the broad acceptance of OSS in this market segment is grounded to the history of software development. In the early days of software development in the 60s and 70s, software was

generally not seen as having value in itself (Fogel, 2005). There was a culture of sharing software between academic researchers and corporations, with a mutual consensus that all actors benefited from this (ibid.). This was at the time when the protocols in which internet is based on was developed, and researchers with an academic background were central to this process (Kogut and Metiu, 2001). Several services originate directly from academic environments. Examples include the Berkley Internet Name Domain (BIND) software which is a Domain Name Service (DNS) and the HTTP protocol, which power the World Wide Web, that was developed at CERN by Tim-Berners Lee in the early 90s (Kogut and Metiu, 2001).

To illustrate how wide spread the use of OSS the server side is we may turn to an example of a web browsing session. The user who starts browsing is about 15% likely to use the open source browser Firefox (W3, 2007a). Upon being able to contact the web server, the browser will use a DNS service to translate a domain to an IP address. This request is 70% likely to be handled by the mentioned OSS BIND service (Moore, 2004). When the IP address is retrieved, there is more than a 50% chance that the web server running at that address is the open source Apache HTTP server, which is a product of the Apache Software Foundation (ASF) (Netcraft, 2007).

A reason for the lack of proprietary software in the early years of software development is related to a lack of hardware standards (Fogel, 2005). The diversity of such standards gradually gave way for a few winners and during the 1980s high level programming started appearing (ibid.). High level programming made it possible to write more general software that could be translated (compiled) to different platforms, hence proprietary software emerged (ibid.). Even though OSS has a strong position on the server side and in academic environments (Kogut and Metiu, 2001), it is struggling with its appeal to the general population with a market share of only 0.6% (Linux) against 97% for commercial alternatives on the client operating system market globally (MarketShare, 2007).

2.10.2 Appeal

One of the factors considered a challenge for the OSS is the general esthetics of the software compared to its commercial rivals. An example of this is the KDE project

(KDE, 2007) which may in many respects be as technically advanced as its commercial rivals, but in a general visual comparison it will for most users appear a bit esthetically challenged. In traditional "geek/hacker culture" a GUI is to some degree considered unnecessary, and many still swear to the command line approach. Less focusing on marketing and "nice wrapping" of the software packages compared to the commercial competition does arguably show, but there are indications pointing to that this is changing.

A shift of focus seems to have emerged in this field over the recent years. After the introduction of 3D functionality in mainstream operating system through Apple's Mac OS X and later through Microsoft's Windows Vista, this has caught the interest of the OSS community. The projects Beryl and Compiz both add visuals at the level of the commercial competition, and now they are joining forces (Linuxlookup.com, 2007). The focus on 3D hardware reveals one key area of conflict between the open source community and commercial actors, which weakens the appeal of OSS to the general population. Several commercial actors, like the 3D chip manufacturer ATI, are reluctant to release open source drivers for Linux for all of their devices (DriverHeaven, 2007). The availability of such drivers is a prerequisite for devices to work properly under operating systems, and the development of device drivers benefit from support of the companies that develops the devices. OSS driver alternatives are often available as part of the Linux kernel, in fact half of the Linux kernel source code consists of device drivers (Robles, 2004), but the lack of support from commercial actors on this subject is an issue that hinders the broad appeal of OSS.

2.10.3 Communities

Edwards (2001) points to OSS development as a learning process, where the involved parties contribute to and learn from the community. He applies epistemic community theories, which implies that all actors with the four characteristics of shared principle beliefs, casual beliefs, notions of validity and a common policy enterprise (Edwards, 2001).

"Contributors rely on a common frame of reference (the four characteristics) as a mechanism for co-ordinating their efforts when venturing into OSS development. This is not a conscious reliance, but a consequence of trying to be part of the community. It is not possible, given the limited means of communications, to socialise people into the community and create that shared frame of reference, which that has a significantly different mindset. The effort to accomplish this task would be far greater than the benefit. OSS contributors are motivated by need and interest in solving the problem, and far less by socialising." (Edwards, 2001, p. 18)

Learning is identified as a process of becoming a practitioner and people who learn are in the process of becoming an important part of the community (or insiders) (ibid.). During this process they actively make contact with community and gradually form their own understanding of the community and its frame of reference (ibid.).

Additionally, open source projects employ governance structures which approve functionality and delegate responsibility (Kogut and Metiu, 2001). Fogel (2005) identifies two common leadership styles in OSS projects, the benevolent dictator and consensus based democracy. The benevolent dictator is a system where all major decisions are made by one person, whereas in the consensus based democracy participants in the community are given voting rights (ibid.). A key function of the governance structures is to avoid "forking" of projects, where a project is fragmented into several, and possibly competing, versions (Kogut and Metiu, 2001).

2.10.4 Development models

Eric S. Raymond is a central actor in the open source movement and the establishment of the OSI. His work includes a very influential essay about OSS development, named "The Cathedral and the Bazaar" (Raymond, 2000). In this essay he discusses two different development models for software development. The cathedral where the software is developed in a closed process with the source code released upon completion and the bazaar model where the entire development process is openly available to the public. He supports a bazaar model with rapid releases, and argues that this is possible to achieve

also in larger development processes. Further, he makes his case by using the success of the frequent releases of the Linux kernel, and his own contributions to several OSS projects. The most famous quote of the essay is "*given enough eyeballs, all bugs are shallow*" (Raymond, 2000), which implies that availability increases bug discovery and that complexity can be tamed by this. He emphasizes this as a major advantage of OSS development, contra the more closed cathedral model often used in commercial projects (ibid.). These arguments proved to be very influential in the OSS scene, which have proven that this somewhat chaotic approach can be highly successful. When it was published by O'Reilly in 1998, it was the first book to be commercially distributed under an open source license. OSS projects have proven to have a growth rate beyond what's normally seen in commercial projects, but despite this the average developers is only putting a couple of hours a week into the development (Robles, 2004).

2.10.5 OSS in developing countries

The appeal of open source in the context of developing countries often lies in its ideology and openness (Weber, 2003). Closed commercial solutions may not satisfy some governments in terms of security and control, whereas OSS can provide total control. Reinventing the wheel by developing their own solution from scratch may in many cases not an option; hence by choosing OSS governments can use existing software while staying in control. With such an approach governments can develop local competence in the software, which can contribute to less reliance on external support and a boost for their own economy (Weber, 2003).

"Open source software should be seen then as more than simply a different kind of product. It is a different kind of process for building, maintaining, and changing the rules that govern information flows." (Weber, 2003, p. 29)

Hence, to regard OSS as just a less expensive alternative to proprietary miss important aspects of that OSS enables. Weber (2003) identifies the motivational factors of developing countries adoption of OSS in three loosely grouped clusters:

Independence: By not locking into commercial solutions, governments can exercise increased control over their own technology. Not only in terms of the mentioned security, but also over the costs and licensing scheme. The price of maintaining proprietary technology can be very high, and this money often goes to companies based in western countries. OSS allows for local competence to be developed and the maintenance to be carried out locally, building up the local economy. Having such competence can be a competitive advantage for developing countries, as they tend to have a lower cost of labor than western countries.

Security and autonomy: A government is required to provide equal service to the entire population. In order to guarantee national security, a government must at a fundamental level be in control over its own information infrastructures. OSS tends to use openly available standards, whereas commercial solutions often use closed proprietary formats. Tying the governmental information to a format provided by a single provider does raise some fundamental questions in regard to self-governance and autonomy of a nation. This is also an issue of ownership and national pride, developing nations desire to articulate their own needs and be a part of the innovation process. Additionally, it is argued that OSS provides an increased level of security over commercial alternatives (Chelf, 2006), primarily because of the amount of programmers involved and the rate in which bugs are fixed.

Intellectual Property Rights and Productivity: An increased focus on property rights at an international level has lead nations to move away from piracy, which makes governments turn to other alternatives. However, the strength of OSS does not only lie in its arguably lower price. The development of OSS is free in the sense that it is not limited by licensing or other economic aspects, and can hence more freely foster innovation and diffusion. Developing nations can participate directly in the development process, and influence the software design to suit their needs. This may decrease their need to adapt software developed and "handed-down" to them by the western world, and produce software that are more suitable for their cultural context.

2.10.6 Localization

As most OSS projects are collaborations over the internet, the term global is close to implied in the context of localization in OSS. The lack of bureaucracy, restrictive licensing schemes and its availability (Weber, 2003) contributes to make OSS ideal for localization. One prominent example of localization of OSS is the KDE project (KDE, 2007), which currently is translated to more than 70 languages. They have guidelines for translators publicly available and anyone is free to join the project and commit localized data to a central repository. Localized data is tested before it's included in a public release, however, this is an example of how open OSS projects are and how the process of localization can be handled in the context of OSS (ibid.). Communication is to a large extent encouraged by easy to use and informal communication channels such as mailing lists, forums and Internet Relay Chat (IRC).

The open source literature on internationalization and localization is usually strongly related specific software packages and, from a community of mostly developers, very pragmatic and technical in nature. This is reflected in the mentioned KDE (2007) and the introduction to internationalization from the Debian project (Kubota, 2006). The progress of OSS localization in developing countries varies, and governmental sponsorship is regarded as an important contributor in order to develop software, such as input systems that implements international standards (Wikibooks.org, 2007).

The limited type of communication channels and especially the lack of face-to-face communication do also have negative effects. Fogel (2005) points out how this contributes to make it difficult for cohesive and dedicated groups to form, and that this lack of dedication hence lead to people switch between projects quite often. This observation illustrates an important difference in the localization process of commercial software and OSS; while OSS usually has to rely on community support commercial projects are translated using prearranged contracts.

2.10.7 Licenses

OSS licenses can be separated into two main categories, the protective and the non-protective. The differences of these can be exemplified by the use of the General Public License (GPL), also known as “copyleft” (Weber, 2003), from the FSF camp and the Berkley University developed BSD license. They both allow copying, modifying and redistribution of the source code. The main distinction between the two is that the protective GPL license requires code to be redistributed under the same license; whereas the non-protective BSD license basically lets the users decide what to do with it at their own risk. In other words, a project that doesn’t want its code used in a commercial project should use a GPL style license (Fogel, 2005). It can hence be argued that the GPL license actually is less free than the less restrictive licenses such as BSD. The reason for this restriction is based on the FSF’s moral belief that all information should be free, and that everyone eventually will benefit from this (ibid.). The MIT/X license is a widely used license that is very similar to the latest revision of the BSD license. Earlier revisions of the BSD license contained a clause that forced users to acknowledge a specific organization making it incompatible with the GPL license, but this has been replaced by a clause relieving the organization of responsibility of the product and prohibit unauthorized endorsement (Fogel, 2005).

Several other variations and nuances exist between OS licenses, but the distinctions mentioned are the most important aspect in regard to localization capabilities. A protective license can be a decisive factor here. As argued by Raymond (2000), availability increases bug discovery, and some actors may not want all bugs discovered. Governments that modify source code for localization purposes such as security may for instance be reluctant to disclose the source code. It should also be mentioned that the license may be applied to other localization data of projects as well, such as language translations, format patterns and local configurations. By separating the localization data it may in some cases be possible to use a different licensing scheme for this aspect, but this will depend on the licensing policy and code structure of each project individually. Localization can in some projects be seen only as a configuration of a specific solution

and not a part of the main artifact. Internationalization solutions are however as any other code likely to be in the scope of licensing.

2.10.8 One Laptop Per Child

With the Moore's law (Wikipedia, 2007g) being more or less applicable in the recent decades, the amount of processing power found in computers today's vastly exceeds what's really needed for most users in their everyday use. The One Laptop Per Child (OLPC) project, which emerged from MIT, is a concept that tries to take advantage of this performance increase to produce a minimal, yet functional computer for education in developing countries. The OLPC project is not actually an open source product, but it relies heavily on OSS. It's often referred to as the "100-dollar laptop", as the production cost of one unit is expected to eventually get down to $100 (OLPC, 2007b).

Localization plays an essential part in it in the OLPC project. In a global perspective of internationalization and localization in OSS development, this is a project that arguably should to be considered. The OLPC initiative is interesting to the global OSS community just by the sheer numbers of potential users and the attention this project is getting. I will hence devote this section to a review of this product, and how it may position itself in the market if it is released. They have a very interesting and pragmatic view of internationalization aspects:

"The OLPC initiative, by its nature, requires international involvement and participation. Developers must keep in mind the broad range of cultures and languages that the laptops must transcend. In particular, activities should not depend on western icons and modes of thinking, but should abstract ideas to a level that would be familiar to humankind in general, where possible. For instance, consider the camera button on the keyboard. Though one might be inclined to label this key with a small image of a camera and lens, the eye graphic speaks directly to our human capacity for vision, providing a cross-cultural icon that represents the computer's ability to capture what it sees." (OLPC, 2007a)

This quote is from the interface guidelines of the OLPC software, which is primarily intended for all software developers who work on the project. By appealing to basic human emotions, they emphasize simplicity in the design to combat the vast complexity of target cultures. If it's released in western countries it will be sold at twice the price of what they will charge in the developing countries, directly financing one computer for a developing country (OLPC, 2007b). This strategy is by the project referred to as the "Buy One, Give One Free" strategy, and the OLPC project further encourages the establishment of direct communication between the western buyer and the receiver of the laptop in a developing nation (ibid.). It's emerging as an innovative product with multiple uses, such as the ability to light up a room that lacks a power source and to be charged manually with a crank handle (ibid.).

Figure 3: The OLPC (www.laptop.org)

This is the first initiative of this scale to provide computing power to developing countries, and now it has competition from commercial actors. Its being debated if this is something that benefits the developing countries or not, but the introduction of commercial actors have been tough to handle for the OLPC project. They will need at least 3 million orders to start production, and they have very few guarantees so far (Bergstein, 2007). Nicholas Negroponte, who is the founder of the project, is currently marketing the OLPC in developing countries and he is confident that they will get the amount of orders they need (ibid.).

Intel develops a low cost computer named the "Classmate", which is by many seen as a direct competitor to the OLPC project. They recently stepped up and donated 1000 of these computers to the school system in Vietnam (VNS, 2007), which indicates just how serious they are at penetrating this market in developing countries. The fact that the OLPC uses an AMD processor may be a motivational factor for Intel, as they represent their most serious competition in the processor market (Bergstein, 2007). Negroponte has been highly critical to Intel's approach and emphasizes how much this commercialization hurts his own project and how it undercuts the philosophy of the OLPC (Sherriff, 2007). Intel CEO Craig Barrett response was:

"Someone at Intel was comparing the Classmate PC with another device being offered in the marketplace. That's the way our business works. I see plenty of opportunities for the two organisations to work together." (Sherriff, 2007)

Intel does of course mean business when entering this market, but Barrett may have a fair point here. The potential market is huge, and to think that it would have only one actor is probably unrealistic. In July 2007 they seemingly "made peace" as Intel joined the OLPC board and will contribute with both money and technical expertise (Bergstein, 2007). Intel is however continuing their development of the Classmate computer, and the OLPC is still based on an AMD processor (ibid.).

This competition also has an ideological element to it, as the Intel Classmates runs on proprietary software such as Microsoft Windows XP (Bergstein, 2007). Hence it can be seen as a battle between the ideology of the non-profit initiative of OLPC using OSS and the fully commercialized alternative from Intel and its partners. This is however a truth with modifications as Microsoft are trying to get Windows to run on the OLPC, but are hindered by its technical architecture which is not standardized (Bergstein, 2007). The Classmate project is also using and encouraging OSS, and it can be configured with the Linux operating system (ibid.). This point to the Classmate as the most flexible alternative in terms of compatibility with existing software, while the OLPC comes off as a tailored solution with a larger emphasis on innovation (ibid.). In regard to internationalization this means that the Classmate has the flexibility to let the users decide operating system and hence internationalization features, while the Linux based OLPC to a large degree is locked into its own internationalization solutions. In OSS tradition they do however avoid reinventing the wheel and use the open source projects Pango (Pango.org, 2007) to handle script layouts and the Smart Common Input Method Platform (SCIM) (SCIM, 2007) for text input (OLPC, 2007a). The active use of such frameworks are an argument for the importance of the OLPC in regard to OSS localization, and the interest shown in this new paradigm by the industry further substantiates its viability.

2.11 Software development

I will give an introduction to software development in relation to internationalization and localization.

2.11.1 Modularization and object orientation

In the context of programming languages, a module is defined as a language construct that allows a number of declarations to be grouped together (Mitchell, 2003). Over the past 30 years, object oriented programming has become a prominent software design and implementation strategy (ibid.). An object extends a module's functionality technically and provides a uniform way of encapsulating almost any combination of data and functionality, and can in principle be from the size of a single numeric value to an entire

file system implementation (ibid.). Objects were invented through the design of the Simula programming language, and refined in the evolution of Smalltalk. Smalltalk failed commercially, but the concepts were adopted by languages such as C++ and Java (Tate and Gehtland, 2004).

I will in this thesis focus on modules at a higher level pragmatically in the context of GSW. The modular term as used throughout this thesis from this point will denote a to some degree isolated collection of resources that may contain objects as well as other resources such as graphical images and configuration files. Such modularization is widely accepted as a method of isolating the effects of change across contexts in GSW (Herbsleb and Mockus, 2003). Modular design is regarded as a means of coping with the complexity of systems, and a lack of modularization can challenge central control and the support for local variations of a system (Nielsen and Nhampossa, 2005). The "freedom to develop" provided by OSS is a contributing factor to governmental decisions to localize OSS (Wikibooks.org, 2007), and modularization is essential in enabling such freedom. Robles (2004) attributes the growth of many open source projects, exemplified by Linux, to the use of organizational and technical modularization. Effective use of modularization is, according to Robles (2004), a key advantage of open source projects over commercial projects. Modularization is hence widely accepted as a mean of isolating functionality, but in order for the components to interact there will have to be some amount of cohesion between modules.

MacCormack et al. (2006) investigate the dependencies between modules in the open source projects Mozilla and Linux, and concludes that a product design often mirrors the organization that develops it. In this study it is suggested that modules facilitates flexibility, and that a larger degree of modularization is achieved in organizations with a structure that require such flexibility (ibid.). This assertion is further identified as a reason for the perceived view that OSS has a larger degree of modularization than commercial software (ibid.). Based on a study of the refactoring of the Mozilla projects source code, MacCormack et al. (2006) indicate that a conscious choice of increasing the degree of modularization increases the amount of contributors to a project.

2.11.2 Aspect Oriented Programming

While conventional object oriented programming (OOP) works well for the modeling of real world objects, object oriented technologies have problems handling services that broadly reach across many objects, such as security and logging (Tate and Gehtland, 2004). Aspect Oriented Programming (AOP) is a relatively new paradigm that deals with such *cross cutting concerns*. AOP can provide a convenient way of attaching behavior to an object through configuration without forcing changes upon a model, and its ultimate goal is to completely remove cross cutting concerns from the model (Tate and Gehtland, 2004). AOP is often used in combination with OOP for specific tasks such as setting up transaction management. This is the case for the DHIS 2 application, where the AOP features provided by the Spring framework is used. The architecture of DHIS 2 and the Spring framework are introduced in section 4.3.

2.11.3 Cultural factors in software design

The key to internationalization is to be able to extract culture aspects and identify them in a neutral way (Mahemoff and Johnston, 1998). Overt factors can as mentioned normally be handled in standardized ways, which usually implies the use of neutral formats and patterns defined during localization processes. This is often not the case for covert factors, which often require deeper understanding of the system both technically and culturally (ibid.). While overt factors usually can be contained in the presentation layer of the application, covert factors may require modifications to the business layer and the core of the application (ibid.).

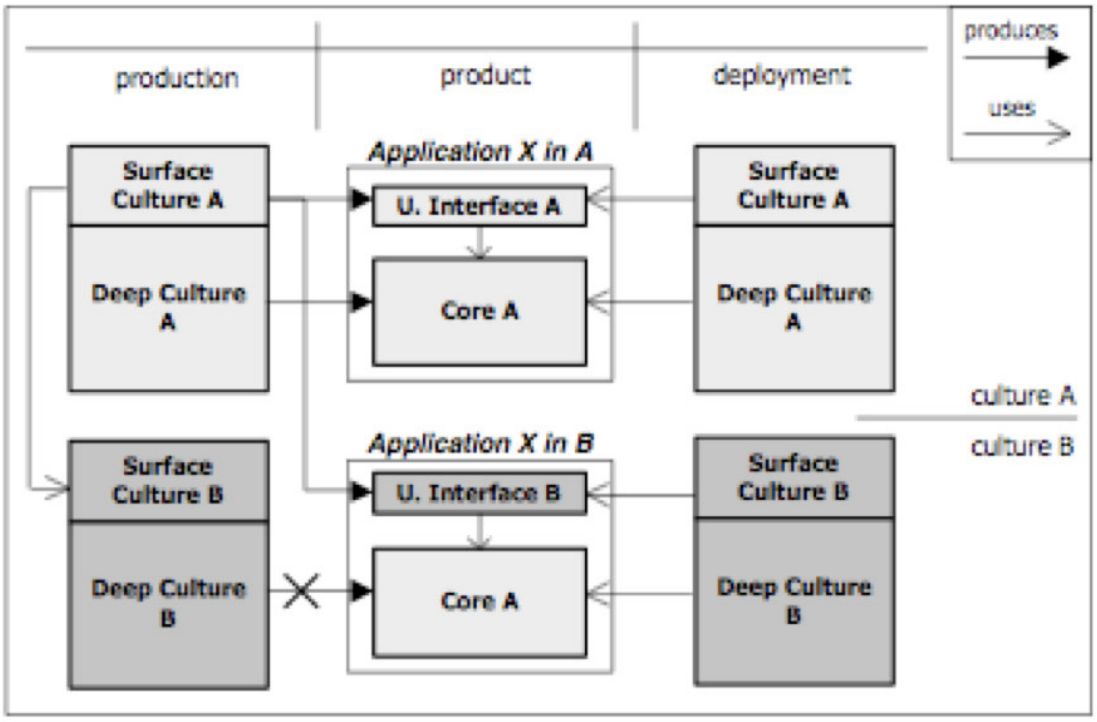

Figure 4: Internationalization of software produced in culture A (Kersten et al., 2002).

Figure 4 demonstrates how cultural aspects can be contained in the user interface of a software design. When the user-interface is replaced in a translation process for target culture B, the core of the application remains the same. Kersten et al (2002) propose a solution where cultural factors can be implemented into the core of applications in a modular way. This concept is presented as *culturalization* (ibid.) to emphasize how it extends traditional internationalization, but this extension is within the scope of the definition of the term internationalization in this thesis (ref. 2.7.1).

Based on a holistic perspective that software by definition is not culturally neutral, the following design is proposed. By determining which software characteristics are culturally dependent, these characteristics can be separated from the rest of the core libraries. It is essentially the same approach as the traditional internationalization, but these, often covert, factors can be more difficult to identify. Kersten et al. (2002) argue that a merger of perspectives from philosophy, sociology and anthropology may be required to capture covert factors in software design.

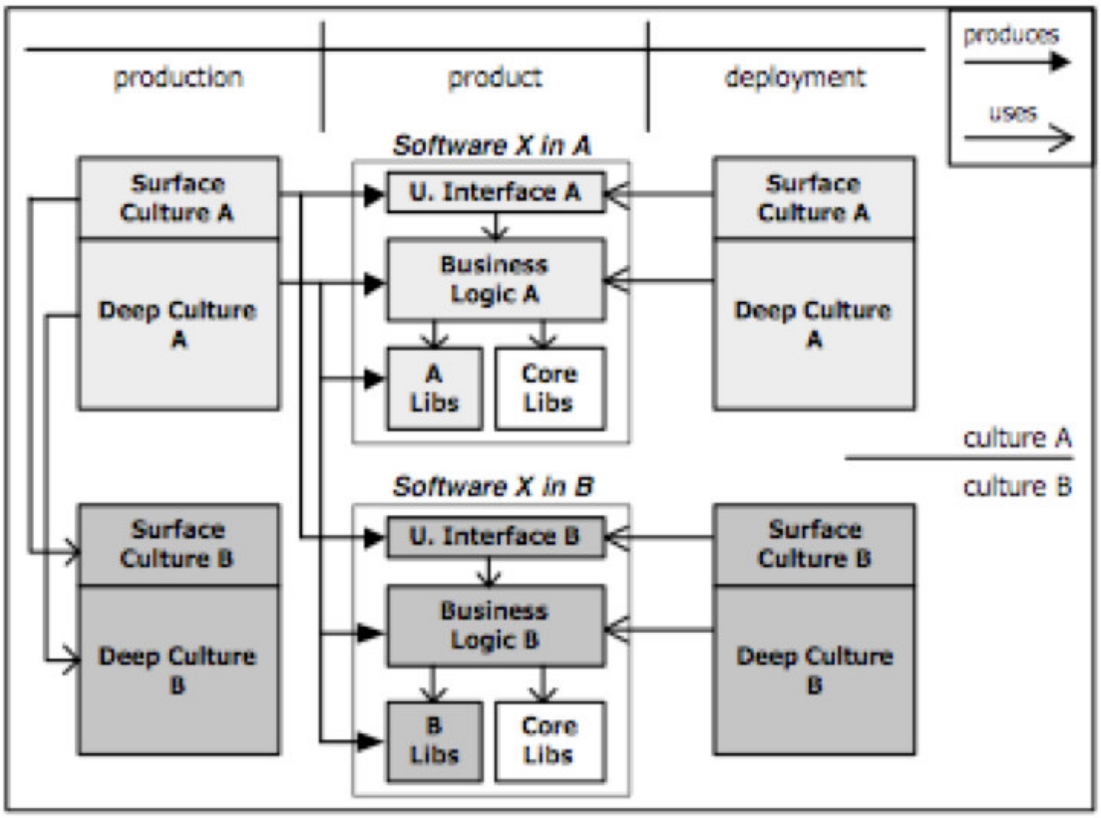

Figure 5: Culturalization of software produced in culture A (Kersten et al., 2002)

By applying the design illustrated in Figure 5, it is argued that deeper cultural considerations will be taken into software design. Kersten et al (2002) further suggest that action oriented programming can be an ideal approach for implementing the cross cutting concerns presented by culturability.

2.11.4 Generification

Pollock et al. (2007) challenge the view that software only can be adapted to local context through a process of localization by focusing on how software can be made to work *across* cultures through *generification work*. Like Kersten et al (2002), there is a focus on separating the generic from the particular, but from more of a reductionist perspective, and with a greater emphasize on the social process of creating generic solutions. By aligning the interests of customers during the requirements engineering, Pollock et al. (2007) demonstrate through case studies that it is in fact possible, to a large extent, to produce general solutions for complex systems. The process that causes complex systems

to have a resistance against translating into other contexts is identified as *particularization* (ibid.). Hence, particularization is at odds with reuse of functionality and generality. Generification work often presents a dilemma of striking a balance between encouraging user's demand for new functionality and diversity, and at the same time striving towards generic solutions. Pollock et al. (2007) demonstrate how general solutions can be developed by capturing collective requirements through a process of aligning roughly similar individual requirements. The process of generification can also be applied to isolate larger regions, for instance continents who share common properties. This approach is referred to by the term *poly-generic* by Pollock et al. (2007), who further concludes by emphasizing the value of investigating system in more detail through *black-blobs* instead of the commonly used *black box* approach.

The process of generification, as defined by Pollock et al. (2007), goes beyond my definition of internationalization, as it advocates the general production of generic software through aligning requirements. The focus on generality and abstracts do however have a lot in common with internationalization, and the neutrality required by the development of internationalization solutions can arguably benefit from keeping a perspective of generification. Additionally, the focus on similarities between cultures rather than just the ability to enable localization, can lead to elegant general solutions that all altogether decrease the need for internationalization solutions.

2.11.5 Graphics

The emergence of globalization and an increasing expectations from the users for better graphics in computer software (Mayeur, 2007), has made graphics subject to many of the requirements that have been presented in this section. Advantages of localizing images are supported through a study carried out at IBM. The majority of users consistently rated a product with superior appearance and visual design as less error prone, faster, and more satisfying than the competitive products, even though empirical evidence of their performance as measured in a formal usability evaluation showed a clear advantage for the comparison products (IBM, 2007). Additionally, the study concluded that cultural effect of images may have a negative effect of up to 15% of a user's willingness to learn

to use an application or web-site (ibid). The awareness of such subjects is reflected in the following quote from the development guidelines from IBM:

"Graphical interfaces and, to a lesser extent, audio visual interfaces are already in existence today in software. Cultural norms influence almost every aspect of a graphical image from color, to shape, to image content elements. Such an interface, when designed with strictly a U.S.audience in mind whether intentionally or unintentionally, will work very poorly with customers in other countries. The nature of the Internet makes it is easy for non-U.S.audiences to view graphical images intended for the U.S" (Mayeur, 2007)

When localized text and static graphics needs to be combined in a picture there is a lot to gain by the use of layering (Esselink, 2000). This can for instance be the case for graphical headlines or product names spelled differently in different countries. By abstracting the overall graphics and style of the text from the textual content, the text can be left easy editable for each localization process (ibid.). For web-based software projects this can be achieved in two distinct ways. The first option is to use image editing software with layering support such as Adobe Photoshop, which will result in static localized images (ibid.). A different and more dynamic approach is to use Cascading Style Sheets (CSS) (W3, 2007b) technology to position text on the image itself at runtime by the client browser.

2.11.6 Unicode

Unicode is a uniform way of defining characters that is designed to work for any platform, program or language. In computing characters are identified by unique numbers, hence a vast character space is need to support all characters. Unicode provides a framework of a scope that can cover all of the characters in the world (UnicodeConsortium, 2007). It is a widely adapted standard throughout the software industry, with support from companies such as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase and Unisys (ibid.).

"Incorporating Unicode into client-server or multi-tiered applications and websites offers significant cost savings over the use of legacy character sets. Unicode enables a

single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption." (UnicodeConsortium, 2007)

In single byte encoding, one character is always expressed by one byte. On the other hand, one character may be expressed in one or more bytes in multibyte encoding (Kubota, 2006). A variable encoding scheme has been adopted to minimize the amount of space required to store Unicode characters, ranging from 8 bit UTF-8 to the 32 bit UTF-32 encoding. In relation to Unicode a few key terms should be established (Kubota, 2006):

Character: Character is an individual unit of which sentence and text consist.

Encoding: Encoding are rules in which characters and text is expressed in combinations of bits or bytes. The terms *character coding system, character code* and *charset* are synonymous with encoding. Examples of encoding includes: UTF-8, UTF-16, UTF-32, ASCII, ISO 8859-1, ISO 2022-JP, JP-1, JP-2, KR, ISO 2022-CN, ISO 2022-INT-1, and VSCII

Character Set: Character sets determine a range of characters an encoding can handle. In contrast to coded character set, this is often called as non-coded character set.

Coded Character Set (CCS): Coded character set (CCS) is defined in RFC 2050 and describes a character set where all characters have uniquely defined numbers. There are several national and international standards for CCS, such as ISO-646 and ISO-2022.

Character Encoding Scheme (CES): Character Encoding Scheme is defined in RFC 2050 and used call methods to construct an encoding using one or more CCS. UTF series can be regarded as CES who's CCS is Unicode or ISO-10646. ISO-10646 defines a Universal Character Set (UCS) standard, which Unicode implements and extends by adding metadata such as reading direction.

3. Methods

In this chapter introduce the methods used, and describe how they have been applied in the context of my research.

3.1 Action Research

Action Research (AR) is a research method from the field of interpretive research. It focuses on close collaboration between researchers and practitioners, and it's defined by Levin and Greenwood (1998) as:

"AR is social research carried out by a team encompassing a professional action researcher and member of an organization or community seeking to improve their situation. AR promotes broad participation in the research process and supports action leading to a more just or satisfying situation for the stakeholders" (Levin and Greenwood, 1998)

This combination of theory and practice allows social aspects to be investigated, which is not as directly accessible using many other research methods. A researcher may by the use of AR attempt to change a system by direct participation, in addition to observing this process. AR is a cover term for several research approaches that have emerged from different traditions. Some see the goal of action research as improving the practice or developing individuals, whereas others see its goal as transforming practice and participants (Herr and Anderson, 2005). Kurt Lewin was the first to develop a theory of AR that was accepted as a respectable form or research in the social sciences (ibid.). In contrast to many other research methods the researcher in action research may make no attempt to remain objective, and can openly acknowledge personal bias to other participants (O'Brien, 1998).

"Action Research is more of a holistic approach to problem-solving, rather than a single method for collecting and analyzing data. Thus, it allows for several different research tools to be used as the project is conducted. These various methods, which are generally

common to the qualitative research paradigm, include: keeping a research journal, document collection and analysis, participant observation recordings, questionnaire surveys, structured and unstructured interviews, and case studies" (O'Brien, 1998)

AR as a case of "learning by doing" can typically be described by a 4 stage cyclic process that consists of: plan, act, observe and reflect (O'Brien, 1998). A problem is identified and a plan of action is created, the plan is carried out, the results are evaluated and if deemed not satisfactory the cycle is repeated (ibid.).

3.1.1 Action research in IS

One important aspect of AR is how it democratizes research by including local researchers and stakeholders. The stakeholders are included throughout the whole process involving defining the problems to be examined, learning and executing social techniques, taking actions and interpret the results of the actions performed (Levin and Greenwood, 1998). Avison et al (1999) describe how AR has gained acceptance in the field of IS and been accepted to be equal in value to quantitative approaches. AR encourages researchers to experiment through intervention and to reflect on the effects of their intervention and the implication of their theories. It may hence be regarded as a less precise approach than quantitative methods, where hard facts are provided as answers. Action research does however allow researchers to observe as well as change complex social situations, for instance the inner workings of organizations.

AR is particularly useful in the research areas of the HISP project, where local empowerment is a cornerstone principle. The culture in the developing countries may in many cases be at odds with the democratic nature of the AR approach with strict hierarchical social structures, and in such cases the researcher may through action research influence the process towards the direction of local empowerment. The relationships built on a personal level during the close collaboration in AR projects can also be of significant value for the researcher in terms of the validity of the research. In HISP research, the local stakeholders are usually the problem owner and the researcher the technical expert of the software that is designed to solve the problem. Combined with

the need of the researcher to be familiar with the local context, this makes the actors interdependent of each other – often fostering the close collaboration encouraged by AR.

3.1.2 Problems with action research

Avison et al (1999) point to the importance of action researchers to be explicit about their research. During action research a researcher assumes the goal of both a researcher and a participant in a project, and the research element should be clearly defined (ibid.). It is also essential for the researcher to have enough knowledge about the research domain. A lack of such competence can lead to a study more similar to a case study, where actions are not properly observed and the researcher relies too much on what the actors say and not what they actually do (ibid.).

"Explicit criteria should be defined before performing the research in order to later judge its outcome, as should ways to manage alterations in these criteria as part of the process of problem diagnosis, action intervention, and reflective learning. Otherwise, what is being described might be action (but not research) or research (but not action research)." (Avison et al., 1999, p. 3)

Avison et al (1999) also point to how the close collaboration implied by action research can lead to personal conflicts, and how action research projects are unlikely to succeed under such conditions. Ethical frameworks can vary between people and cultures, and Avison et al (1999) emphasize the need for researchers and practitioners to establish and share a mutually acceptable ethical framework. Establishment of collective requirements, level of participation, confidentiality agreements are ethical considerations that can provide a framework to prevent conflicts, along with a common sharing of information generated by the research and early disclosure of personal biases and interests (O'Brien, 1998)

3.2 Survey

Survey as a methodology can be both qualitative and quantitative. Qualitative survey methods started to gain prominence in development projects during the 1980s, primarily

in response to the drawbacks of quantitative questionnaire-type surveys, which were considered time-consuming and expensive and not suitable for providing in-depth understanding of an issue. This led to a polarization between the "formal" quantitative and the "informal" qualitative methods. During the 90s attempts were made to highlight the complementarity of the two approaches and how they can be combined (Marsland et al., 2001).

Surveys meld sampling, question design and data collection methodologies to produce statistics. Fowler (2002) presents three main areas commonly associated with surveys:

- Measurement of public opinions for newspapers and articles.
- Measurement of political perceptions and opinions
- Market research designed to understand consumer preferences and interests.

Breakthroughs in computer technology are also an essential factor in the use of surveys. The computing power available to analyze the data and the ease of conducting surveys over the internet is advantageous, especially for formal quantitative approaches in which parts of such processes can be fully automated.

3.3 My research approach

My primary research method is AR, which is reflected by my direct involvement in the research from the beginning. Starting out the research was as mentioned mostly technical getting an overview of the technology and the DHIS concepts. This background was essential to have in order to perform the next step of field research, which included both development and implementation tasks.

The nature of the action research approach of the HISP project allowed us to develop the internationalization solution along with local tech staff and to test it on the real users in the health sector. This study draws close to being a case study of the localization for Vietnam for two main reasons. The first internationalization solution was developed in Vietnam and the application was first fully localized for Vietnam and Vietnamese. It was however later localized for several locations in India and in African countries and I

include these experiences for a global perspective. The challenge of internationalizing the system for India did pose specific new challenges, for instance the leap in character system was much bigger from the common Latin-based systems of both Vietnamese and English. Additionally, a technically challenging case emerged from India, which is presented in section 4.5.

I have had the opportunity to observe the project from a variety of different angles. From a role as a developer, team leader for an implementation and a group teacher for an introductory course to the project. I believe that these perspectives have been a strength my view of the project, but there will always be a risk of missing important aspects which leads to me to my secondary research method.

3.3.1 Survey

My intention was initially to use the survey to get an impression of the global localization effort in DHIS 2. It did however prove to be difficult to get a satisfactory sampling from this for a pure quantitative analysis. Lack of resources, lack of internet connection and somewhat paradoxically language problems seemed to be the reason for this. Instead, I decided to use a more qualitative approach in my survey design. Limiting the survey to the Vietnamese and Indian implementations and asking for specific comments rather than only quantitative data. The main goal of the survey is to get an overview of the level of satisfaction among the users of how well the DHIS 2 application translates to local contexts. Focusing on the localization capabilities generally and the internationalization features implemented by me specifically, the whole survey is included in Appendix C.

By doing this I hoped to uncover new issues related to the localization process on the "ground level", particularly issues that have surfaced after my field trip. Through the use of action research it was highly possible that my own assumptions and personal biases at times overshadowed actual facts, and I hoped that a survey could point me to such cases and if necessary provide balance to improve the validity of my research. The survey conducted is also supplemented by direct interviews by email and IM. During my time in Vietnam I became good friends with several of the people surveyed and interviewed,

which can have both positive and negative effects. I do think that this has been helpful in terms of getting legitimate answers, but I it's difficult to verify this.

4. Case

In this chapter I present background information for the case in which I carried out my empirical study.

4.1 HISP

HISP was founded in 1994 after the fall of Apartheid as a collaborative effort between researchers in South-Africa and Norway. The new and democratic South-Africa inherited one of the least equitable health care systems in the world, with 60% of the resources serving only 20% of the population (Braa and Hedberg, 2002). There was a serious lack of common collection methods and standards to cover the entire population. By working with the government, the HISP project was through pilot projects able to establish such routines and standards. Braa and Hedberg (2002) is commonly referred to as the manifest document of the HISP project. It lays out the core philosophy and principles in which the project is founded on and describes the conception of the project itself.

The primary goal of the HISP project is to design, implement, and sustain Health Information System (HIS) following a participatory approach to support local management of health care delivery and information flows in selected health facilities, districts, and provinces, and its further spread within and across developing countries (Braa and Hedberg, 2002). This is encompassed by three goals:

- HIS design, development, and implementation, including improved use of information.
- Organizational and human resources development.
- Developing theoretical and practical knowledge.

At the core of HISP is a philosophy of empowerment at all levels of the health hierarchy. A key to achieve such empowerment is to enable all levels in the health hierarchy to define their own standards as long as they implement the standards on the level above. This philosophy is visualized in the hierarchy of standards (Braa and Hedberg, 2002):

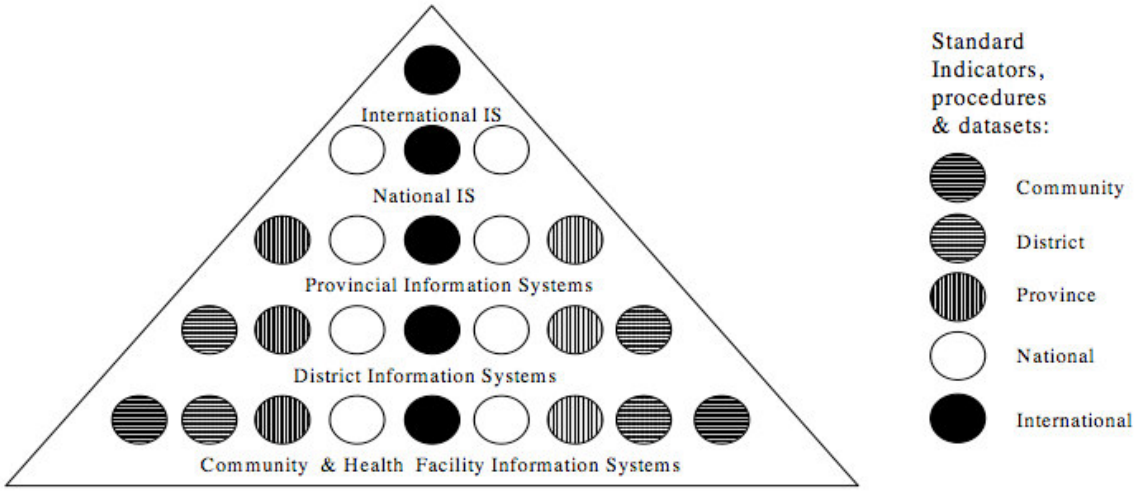

Figure 6: The hierarchy of standards (Braa and Hedberg, 2002)

The hierarchy of standards stands in contrast to many commercial projects, where standardization is used as a means of control to enable coordination from a central management. This is reflected in the case from the shipping industry in Rolland and Monteiro (2002), where centralized "strict standardization" is suggested by the management as the only viable alternative to global standardization. HISP encourages the development and use of standards locally, with the only requirement that every level conforms to the level above in the hierarchy. The hierarchy of standards was an essential "attractor" for HISP in the effort towards standardized data sets in South-Africa (Braa et al., 2007). As it is a flexible standard, that both enables and encourages flexibility at all levels, an agreement with local health stations on a common and minimal data set was reached (ibid.). This was a breakthrough for HISP, and its concepts and technology has since diffused.

After the success in South Africa, HISP has extended its research to a series of other countries including Mozambique, Malawi, Tanzania, Ethiopia, Nigeria, Zanzibar, Vietnam and India. It was also established in Cuba for a short period, but that initiative got shut down by the government. HISP can be conceptualized as having a relatively loose and modularized project structure. It's comprised of a horizontally structured level of countries, where each country has their own vertical level of various institutions and

governmental agencies (Braa et al., 2004). Each country node can be structured differently and implementation efforts are usually funded by international donors and local health authorities. The nature of the nodes is different from country to country and their respective local contexts, but they are usually able to operate quite independently from the rest of the HISP project (ibid.). One key role of these nodes is to provide translations of requirements from the ground level, this is crucial as local health workers very rarely are proficient in English. Funding of the research activities within the HISP network, such as scholarships for doctoral and master students, are primarily done by the Norwegian government (ibid.).

4.2 DHIS

The district health information software (DHIS) is the main tool used by the HISP project. I will first describe the central aspects of the application, and then the two generation of products that implement these aspects. The philosophy of HISP is reflected in the core functionality of DHIS. A participatory approach has been used in the development of the software itself, with medical staff closely involved in the development process. The design of the DHIS software prescribes decentralization and local analysis and use of data (Braa and Hedberg, 2002).

4.2.1 Concepts

The concepts described below are all dynamic data, in the sense that they can be changed by the users while the application is running. This is a key characteristic of the DHIS that aligns with the overall goal of the HISP of empowering local health facilities.

A hierarchy of organization units is defined, reflecting the real world health units. The data entered are organized in datasets, which contain data elements. Data is entered for defined time periods, and one specific value entered for one data element for one period for one organization unit is referred to as a data value. Data can be exported and imported within the organization unit hierarchy and this is usually done monthly or quarterly. Health indicators can be defined containing a numerator and a denominator, which again basically contains formulas with references to data elements and arithmetic operators.

Combined with aggregation of data in the levels in the health hierarchy, the indicators present a powerful feature. Data can be aggregated based on the sum of all values for a data element of the underlying organization units or an average. Data that rarely changes, such as the number of beds in a hospital, is referred to as semi-permanent data and the data collected routinely is referred to as routine-data. The last type of data collection is survey-data, which for instance contains information about patient satisfaction.

Data is presented to the users in reports that can be designed to resemble the physical reports used before the system was introduced. These reports may contain aggregated data and calculated indicators, and they can be customized at all levels in the hierarchy where the software is implemented.

4.2.2 DHIS 1.x

The first version of DHIS was implemented and used to capture and analyze monthly data at district, regional and provincial levels in Western Cape (South-Africa) in 1998 (Braa and Hedberg, 2002). DHIS was first developed as a typical action research experiment, using rapid prototyping to support the implementation of data standards at pilot sites. As the user base increased the development of DHIS 1.x turned into an ongoing evolutionary software development project. The prototyping strategy was however kept by applying it to a few selected sites before releasing new versions to all users (ibid.).

DHIS 1.x is developed by a team based in South-Africa (Nhampossa, 2004). Along with producing new versions of the DHIS, the team is responsible for providing technical support to the teams engaged in localization in different countries (ibid.). DHIS 1.x is shipped with on an installation CD, which includes a user manual and additional support software tools including various Microsoft service releases/packs (ibid.). All the complementary and supplementary resources available on the DHIS setup CD are in English and for English versions of Microsoft Windows and Office (ibid.). DHIS has traditionally been packaged and made ready for installation by the team in South-Africa,

and then distributed to the different nodes in the HISP network through the regular HISP channels (Nordal, 2006).

HISP has all along defined DHIS 1.x as open-source software, and the license, which is a GPL variant written by HISP, has been bundled with the application (Nordal, 2006). DHIS 1.x is based on the proprietary platforms of Access and Visual Basic and bundled with its own source code (ibid.). The nature of these platforms makes it possible for any user to view and modify it when the application is running, but such changes will only affect the local installation of the application (ibid.). Any changes that should be propagated to other installations must therefore go through the South-African team, which is capable of integrating the changes and repackaging the application (ibid.). During a localization process in Mozambique it was necessary to modify the DHIS application, which with its centralized development structure proved to be frustrating:

"The multiple adapted release cycles of the DHIS software suggested an endless process of interaction (with South Africa), whereby the integration of the new releases implied starting more or less from scratch. The new initiatives and features locally implemented in Mozambique are at the same time not necessarily relevant and even compatible to the new releases generated for internal use in South Africa." (Nielsen and Nhampossa, 2005, p. 7)

DHIS 1.x was not internationalized initially, simply because it was not intended to be used outside South-Africa (Nielsen and Nhampossa, 2005). Hence the initial releases were designed and implemented to meet the language, format, culture and regulation requirements of South-Africa (ibid.). Multilanguage was enabled through a separate database module named HISPML (HISP Multilanguage Library) (Nhampossa, 2004). This module makes it possible to translate DHIS to all Windows 2000/XP supported languages. When the current locale of the operating system is changed, the text strings are automatically updated to the selected locale in the DHIS application (ibid.).

To combat some of the limitations of the initial design of DHIS 1.x, a new version the product was completely refractored using similar technology. The new version, DHIS 1.4, was not as dependent on the Microsoft Office technology used in the previous versions, and it was able to use some external database servers, such as MySQL (Nordal, 2006).

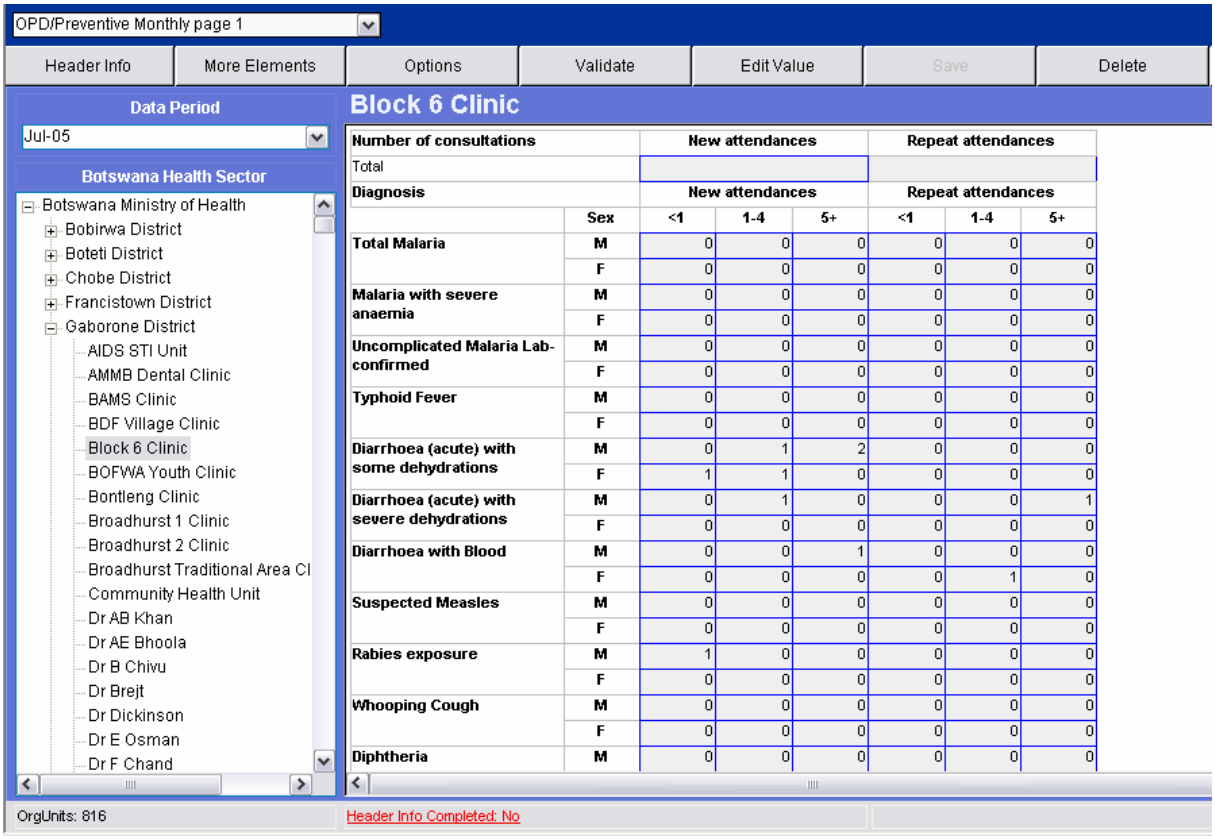

Figure 7: DHIS 1.4 data-entry screen (Botswana)

4.2.3 DHIS 2

DHIS 2 is developed as the successor of DHIS 1.x to combat many of its limitations, both technical and in terms of the distribution of the development process. DHIS 1.x is as mentioned based on Microsoft Office proprietary technology, which locks the users to the Windows operating system and often leads to piracy (Nordal, 2006). DHIS 2 is developed using OSS frameworks only and may legally be run for free. The source code of DHIS 1.x is open source in the sense that anyone is free to modify the program, but the development process is not open and carried out by a team in South Africa (ref. 4.2.2).

The structure of the program has a complex data model that is often referred to as having an “onion” structure, which makes it very difficult for multiple contributors at the same time. DHIS 2 uses a modular structure and tools that facilitate global development, such as a globally available source code repository and communication tools (ref. 4.3.3). One other enhancement over its predecessors is that it’s Web-based. This adds flexibility in terms of the technical deployment, as it can both run as a standalone web application serving one client and as a central server serving multiple clients.

DHIS 2 is distributed under a BSD license (ref. 2.10.7), which allows anyone to do exactly what they want with it. It’s somewhat unclear if all the localized data in the source repository of DHIS 2 is BSD licensed, as only the java source files contain this license. It’s also unclear if this is the official license of all information generated by the DHIS 2 project. Licensing issues have been discussed several times without fully reaching a proper consensus (Nordal, 2006). The documentation of DHIS 2 exists for the most part on an open Wiki (HISP, 2007), but no license is officially defined for this information.

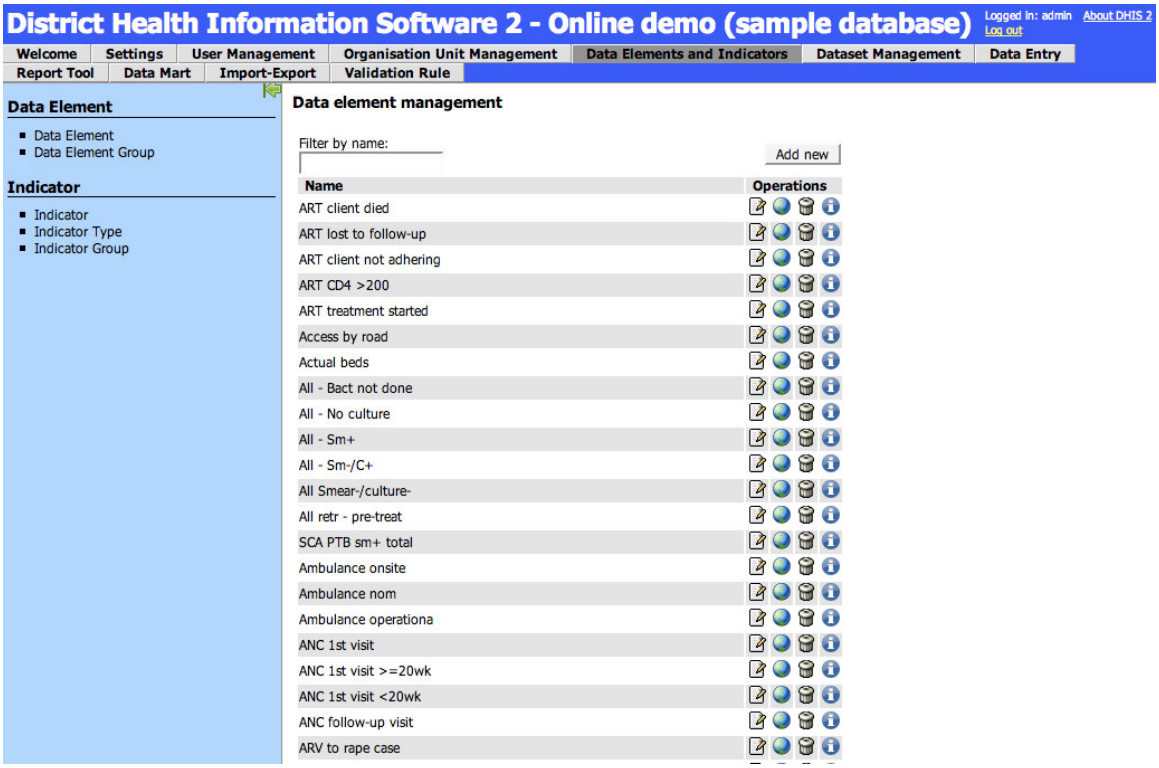

Figure 8: DHIS 2 screenshot

4.3 Technology and project structure

I will in this section describe the technology and the structure of the DHIS 2 project, with emphasis on aspects relevant to my empirical study.

4.3.1 DHIS 2 architecture

A programming practice of separation of concerns is practiced in DHIS 2, which implies breaking down problems into smaller components to make them more manageable and less dependent on each other. The overall architecture of DHIS 2 consists of three tiers:

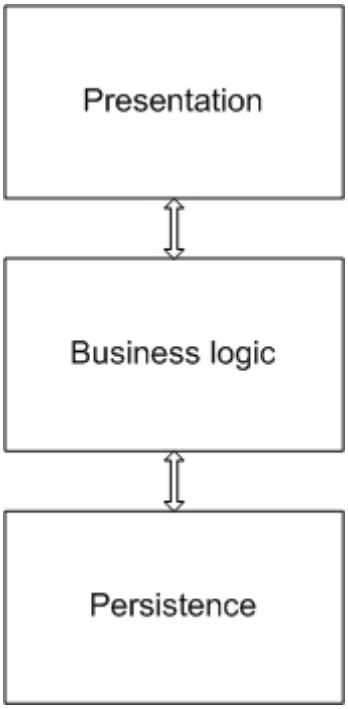

Figure 9: DHIS 2 three-tier architecture

Presentation: The user interface translates information from the business layer into information viewable by the users and back. In the case of DHIS 2 this is a web based system. The web modules can be separated into three different categories. Firstly maintenance who provides manipulation of the data objects, such as adding, deleting, editing and updating. Secondly modules handling output of data, such as the report module and import/export functionality between DHIS 2 installations. The last category contains data entry, which is the module used by medical staff to enter values into the system, and data mart who is a caching solution for calculated data elements and indicators (Øverland, 2006).

Business logic: Performs evaluations and calculations. The use of this tier often involves just dispatching data between the presentation and the persistence layer without any evaluations or calculations. This tier is often referred to as the service layer.

Persistence: Provides an object-relational mapping between the database and java objects. This tier is responsible for retrieving and storing objects. These objects are for the most part plain old java objects (POJO), which implies that they have a quite simple

structure without any complex implementations. This layer is also referred to as the database layer.

4.3.2 Frameworks

DHIS 2 uses frameworks such as Spring and Hibernate, which are considered lightweight in the context of enterprise Java programming (Tate and Gehtland, 2004). They provide an alternative to the complexity introduced by commonly used enterprise solutions such as Enterprise JavaBeans, which increasingly are considered to be too complex (or "bloated") and invasive on the software design for many projects (ibid.).

Spring: Spring is a full-stack enterprise application framework that emphasizes simplicity. It is built as a lightweight alternative to many of the heavier enterprise application framework solutions such as EJB 3. Rather than to try to do everything, Spring focuses on integration with other open source frameworks. By letting the developers choose which services are needed it lets the developer decide what role it should have in projects. It has a layered and consistent architecture with features such as:

- Transaction integration.
- Aspect Oriented Programming (AOP).
- Integration with Object/Relational Mapping (ORM) frameworks such as Hibernate and JDO implementations.
- Its own Model View Controller (MVC) framework, but it also supports integration with other MVC frameworks.

Spring has been an open source project since 2003, and it has 20 active developers (Johnson, 2005). The terms Inversion of Control (IoC) and Dependency Injection describes functionality at the core of Spring, where the responsibility for creation of objects are given to the framework instead of the programmer (Tate and Gehtland, 2004). Good programming practices are encouraged through programming to interfaces, which makes it easier to manage implementations and enforces a strict typing. Bean properties and dependencies set through Dependency Injection in the configuration files are easily readable and configurable. Spring is used consistently between all three tires of DHIS 2,

"wiring" all of the beans (reusable Java objects), their properties and setting up the transaction management.

Hibernate: Hibernate is an ORM framework that maps objects to Database Management Systems (DBMS) based on a SQL-schema. It adds a level of abstraction above the SQL code in a general SQL language named HQL (Tate and Gehtland, 2004). SQL statements are translated to the database systems using supplied dialects, and a vast amount of relational databases are supported through JDBC. Hibernate is also able to generate SQL statements based criteria set through Java code, which often results in increased readability and elimination of SQL syntax errors not picked up by the programming IDE. Referenced objects are not by default loaded until requested as a means to improve performance (commonly referred to as lazy loading). The objects being persisted do not have to implement any special interfaces and can be just POJO, hence Hibernate is often referred to as *transparent persistence* ORM framework (ibid.). DHIS 2 uses Hibernate on the persistence-tier through store implementations to handle object persistence.

WebWork: WebWork is a web framework that is built on top of xWork, which is a generic command pattern framework. It is described by its authors as:

"WebWork is a Java web-application development framework. It is built specifically with developer productivity and code simplicity in mind, providing robust support for building reusable UI templates, such as form controls, UI themes, internationalization, dynamic form parameter mapping to JavaBeans, robust client and server side validation, and much more." (WebWork, 2007)

Recently WebWork merged with a similar framework named Struts, and the 2.2.5 release was its last release. The next release will be a merge of the two projects and it will be named Struts 2 (WebWork, 2007). In DHIS 2, WebWork is closely integrated with Spring and the web projects are module based. A portal project assembles these modules into one package, making it possible to customize what modules to use. This approach also allows single modules to run individually, which helps to reduce the time needed for

testing during development. Velocity is used as the template framework. This is a quite simple yet powerful template engine that promotes good programming practices through a clean syntax and limited support for logical operations (that in most cases should be performed elsewhere).

4.3.3 Tools

Maven: Maven is a software project management and comprehension tool that can be used for building and managing any Java based project. Maven allows projects to be built using a project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Automatic generation of documentation is also supported, along with unit test support and dependency management (ASF, 2007).

The DHIS 2 module hierarchy is created using Maven. This is advantageous to the project as it grows, as Maven handles the dependency management. All project dependencies are available through a remote repository and will automatically get downloaded into a local repository when needed.

Subversion: Subversion is a version control system that aims to replace the aging CVS system (Tigris.org, 2007). At the core of Subversion is the repository, which is a file-hierarchy. Users may remotely do changes to this file hierarchy, and a record of all of these changes is kept. The version control and file management features makes this tool powerful and essential to virtually any software development project that is too big for anyone to do alone.

Subversion plays a very central role in the DHIS 2 development process. In addition to the version control of all the files, it reports all changes done to the repository on a mailing list. Log messages are provided for all changes done by the programmers, making this a basis for further discussion on the mailing list and encouraging collaboration. As of the spring of 2007, the repository itself was also made available over HTTP, making it available in developing countries with limited internet access.

Confluence: Confluence is an enterprise product freely available to open source projects (Atlassian, 2007). Wiki is a concept where all users are encouraged to add, remove and edit the content of web pages. This can be done relatively easily through the web pages, including adding images and linking to other web pages within the Wiki itself or to external web pages. The Confluence Wiki serves as the main documentation site for DHIS 2. It contains a vast amount of information about the project itself and research related to HISP and the DHIS software.

JIRA: JIRA is a sophisticated bug tracking, issue tracking, and project management application. Produced by the same company as Confluence, this product is also commercial software. The source code of the application is provided to paying customers with a strict proprietary license prohibiting further distribution.

Mailing lists: Several mailing lists are in use by HISP and the DHIS 2 project. In addition to the mentioned version control list, we have the following lists: developer, users, hisp, dhis14 and one for service maintenance. Most of the discussions are done on the developer mailing list, and the other lists are very rarely used. The version control list is read-only, so discussions originating from that list are forwarded to the developers list. The users list is aimed at non-technical users, but in practice all the discussions happen on the developers list. The developers' mailing list is archived on a webpage which works well for referencing previous conversations, but the features of this archive is very limited beyond that.

Instant Messaging (IM): This is in combination with the developers' mailing list the main discussion forum of DHIS 2. Text based online meetings are held with participants from all the main countries involved with HISP. Decisions are often reached during such meetings, resulting in roadmaps for future releases that have a common consensus. Voice chats are also used in some cases, but lack of infrastructure in terms of bandwidth in developing countries and language problems often makes this a complicated solution. The IM programs mostly used by Norwegian developers are the Microsoft's Live! service, whereas in Asia the Yahoo! Messenger is the most frequently used. However, for

the “official” online meetings, Microsoft Live! is always used. My observations indicate that IM is the most used communication methods between developers in the project. This is a fast and efficient way of discussing technical programming challenges that are too “small” for the mailing list. However, information that should be made publicly available may be left undocumented by the use of IM.

4.3.4 DHIS 2 setup

Two main components are needed in order to run DHIS 2. The application is as mentioned platform independent in the sense that it supports any platform that is able to run Java Virtual Machine (JVM) version 1.5. Services to be used with the DHIS 2 application may be chosen by the users locally. At the persistence tier any database supporting Hibernate may be selected, this implies that it has a Hibernate dialect translation and a JDBC driver (ref. 4.3.2). A Servlet container is required in order to run the application, which handles the environment of DHIS 2 in terms of memory available and HTTP interactions. Servlet containers are usually very flexible in terms of integration with existing web server solutions. Because OSS licensed versions of both of the database servers and Servlet containers are available for most platforms, DHIS 2 is regarded as platform independent.

4.3.5 Roles

I will briefly explain the structure of the DHIS 2 project. As mentioned DHIS 2 is a product of the HISP project, hence the official leadership consists of three professors from the HISP project. One professor is from India and the other two are from Norway and the UiO. The software development process of DHIS 2 is managed by two PhD students both situated at UiO. In practice, the PhD students manage most of the implementation efforts and they also provide guidance to most of the master students. Below the PhD students, a group of core developers is appointed with extended decision making authority over the development of DHIS 2. The core developers consist of master students and former master students from the UiO. Below this level of authority are the developers from UiO and teams hired in other countries. Most of the hired developers are not participating in the direct development of DHIS 2, but rather doing localization work

and supporting local implementations. This often results in that they assume roles similar to end users through requests of new functionality and technical support.

4.4 Vietnamese context

I will in this section present the Vietnamese context with a focus on aspects relevant to my case study, which to a large degree is based on a 4 month field trip to Vietnam. Some of my own observations about the Vietnamese context are included in this presentation.

4.4.1 Demographics

The capital of Vietnam is Hanoi, but the biggest city and the main focus of this study is HCMC (formerly known as Saigon). The population of Vietnam is estimated to around 85 million and it has an area of 331 689 km^2. I will introduce the essential demographics for HCMC in section 5.2.1 where the implementation plan of DHIS 2 for this city is presented.

4.4.2 Politics and social structure

Vietnam is among the last five countries in the world controlled by a single-party communist system (Wikipedia, 2007a). Although some of the ideology seems lost, Vietnam officially keeps a commitment to socialism. The party in control has appointed governmental officials such as a president with both of the titles of “head of state” and a military “commander in chief”; additionally a prime minister acts as a “head of government”. This is a part of a strictly hierarchical structure, with the prime minister preceding a council of ministers composed of deputy prime ministers and the heads of 26 ministries and commissions (Wikipedia, 2007h).

The military stands strong in Vietnam. It’s highly visible in the everyday life, and generally surrounded by a fair amount of paranoia. For instance, when visiting the home of a Vietnamese friend working in the military, I was not allowed to photograph any military items and I had to promise to not put any of the pictures on the internet. A course named “politics” is compulsory for all grad students, which basically is propaganda from the ruling party. My impression of the political situation is that the people generally seem to accept the situation. As long as they avoid any political subjects, and keep their heads

down, they are free to do what they want. This is by in large understandable, as the Vietnamese government does enforce laws when they feel threatened. Such as the jailing of a priest from the Hue province for speaking up against the government, resulting a 10 year conviction for "undermining national unity" (BBC-News, 2003). My impression is that people are used to being "told what to do", and this can according to my observations to a large degree be attributed to the governmental suppression and the strictly hierarchical structure in Vietnam. The Vietnamese culture is highly influenced by western culture, with international TV channels such as CNN, BBC and HBO commonly available. My impression was that they were friendly and somewhat curious about western culture, but I did pick up a common hostility towards USA.

4.4.3 History

The area now known as Vietnam has been inhabited since Paleolithic times, and the first Vietnamese civilization is linked by archaeologists to the early Bronze Age. It was a part of the Chinese dynasty for about a thousand years until the year 900, and the next 900 years were characterized by division of the country between dynasties and civil war. Vietnam's independence ended in the mid 1800s when it was colonized by the French empire. Christianity was introduced and their natural resources such as tobacco, indigo, tea and coffee were exported. After the first Indochina war (1946-1954), a divided country between the south and north gained their independence through a forceful national insurgency lead by Ho Chi Minh. The US had sided with the French during this war, which eventually lead them into a war against the communist north-Vietnam, and the Viet Cong army. This war ended in a US withdrawal from Vietnam and the Paris Peace Accords on January 27, 1973 formally recognized the sovereignty of the north and south. After a short period the sides were officially reunified under communist rule as the Socialist Republic of Vietnam on July 2, 1976 (Wikipedia, 2007h).

4.4.4 Economy

Due to large economic losses during the war and a North American and European embargo that followed, the Vietnamese economy was in bad shape for several decades. They did have trade partners in other communist countries, but the economy didn't start growing significantly until after a market economy that encouraged private ownership

was introduced in 1986. Vietnam was the second fastest growing economy in the world between 2000 and 2005, and is now the third-largest oil producer in Southeast Asia. It is also an attractive target for information technology, and especially outsourcing, because of the low cost of labor (Wikipedia, 2007h). In the end of 2006 Vietnam was accepted for a membership in the World Trade Organisation (WTO) (WTO, 2006).

4.4.5 Languages

About 86% of the population speaks Vietnamese as their native language, and French is commonly used as a "status language". French is loosing popularity towards English as the second language, and English is now mandatory in most schools (Wikipedia, 2007h). The rise in tourism makes proficiency in English important in terms of job security, as it can provide jobs in this very lucrative industry (ibid.).

4.4.6 Health system

The Vietnamese health systems primary focus, in regard to management of information, is on collection and transmission of data through standardized health forms. Administratively, the country is divided into 61 Provinces and 631 Districts. There are 803 hospitals and 10 293 commune health centers (CHC) (WHO, 2005). 7 health-statistic forms are used at the commune level, 15 at the district level, and 16 at the provincial level. Emphasis is placed on reporting indicators to the higher levels, but not enough emphasis is placed on the use of information and analysis (WHO, 2005).

The general information flow in the health hierarchy, with variations among some vertical programmes, is: (WHO, 2005):
Ward → district → province → Ministry of Health (MoH)

The actual transmission of reports is not always done by post. Especially the reports from the commune health centers are often handed over at meetings. Many are not filed at all, and there are no guidelines on how to handle missing reports when calculating indicators. The resources available are generally considered insufficient at all levels, and too many resources are devoted to software development (WHO, 2005).

4.4.7 ICT

Several steps have been taken by the Vietnamese government to combat the digital divide (ref. 2.3) and to integrate Vietnam into the global economy. The status of their "E-readiness" is an essential part to continue economical growth, especially in regard to the outsourcing industry. In October 2000, an official plan was released by the government to support the following goals (Elmer, 2002):

- To create an enabling environment for the use and development of IT in support of modernization.
- To ensure widespread and efficient use of IT in all sectors.
- To develop the national information network to reach global levels in coverage, quality and costs.
- To develop human resources to support the use and development of IT.
- To develop the IT industry as a spearhead economic sector with an increased contribution to GDP growth.

OSS localization to the Vietnamese language began in 1998, primarily as an individual effort by an enthusiast. It has now grown to a team of between 20 and 50 people and examples of OSS localized for Vietnam include GNU/Linux, KDE, Gnome, OpenOffice.org and Gaim (Wikibooks.org, 2007).

The level of computerization of health data is still very low. Village health workers use paper notebooks using their own page layout, while commune health stations usually have printed standardized registers (WHO, 2005). Øverland (2006) revealed that the assertion by the Ministry of Health that computerization is applied at the district level and above is a truth with modifications. His research from rural areas in Vietnam included several visits to district offices without any computers present. All districts in HCMC are however computerized, but an interview I conducted with an IT administrator at a hospital in HCMC indicated that none of them are using OSS based operating systems. The Vietnamese government has vouched to use OSS for both idealistic reasons and as a measure to fight piracy (ibid.), the latter was an essential credential for their acceptance into the WTO in 2006 (WTO, 2006). I did not observe any use of OSS in any

governmental buildings or universities during my four month field trip, except for software introduced through the HISP project. A lack of willingness towards using OSS technologies was further indicated by the visit of Microsoft founder Bill Gates to the country last year, where deals were signed for licensed Microsoft software in the public sector (BBC-News, 2006).

4.4.8 DHIS history in Vietnam

To further provide a perspective on the context in which or system was localized I will outline the history of DHIS in Vietnam prior to my field study. I will rely heavily on Øverland (2006), who has done an in dept case study of the activity of HISP in Vietnam. This case study includes my contribution to the implementation effort in Vietnam, as well as implementation efforts before and after my field trip.

The first implementation of DHIS 1.3 started in November 2004 after initial contact was made in July 2004. An agreement was reached with a company named *OutSoft*, which was to take responsibility for both specific development tasks with DHIS 2 and implementation. The development side of this was lead by a Norwegian master student, and the implementation of DHIS 1.3 was lead by an OutSoft employee. The development part of the agreement eventually broke down without much results, mainly due to workers being unfamiliar with the technology used in DHIS 2 and communication issues (Nordal, 2006). Eventually the implementation failed as well, Øverland (2006) points to three major causes for this:

- The report generator developed by the OutSoft employee had several flaws that were not resolved. This lead to annoyed users, who eventually stopped using the software. The inability to solve this situation was embarrassing to HISP Vietnam.
- There were technical issues regarding encoding that lead to incorrect rendering of Vietnamese characters. Additionally, the computers where DHIS 1.3 was installed had severe stability problems due to virus attacks.
- Developers from OutSoft were reluctant to do implementation tasks and support work as agreed upon prior to the implementation. They were more interested in doing development work.

At most 17 districts were using DHIS 1.3 to submit data, and the implementation effort faded out in the autumn of 2005 (Øverland, 2006). The second implementation effort, which planned to implement DHIS 1.4, never materialized. It was however very useful as many relationships were built during this process, with a report generator as the most important product. A Vietnamese team was trained and hired, and HISP built a relationship with the MoH and regained important contacts in HCMC and Hue (Øverland, 2006). The initial plan was to first implement DHIS 1.4 and then have a transition to DHIS 2. Many actors, myself included, opposed this as a too complex approach. A partner university was not interested in supporting the DHIS 1.4 solution, which is based on Microsoft VBA technology, and DHIS 2 had matured to be ready for production use through a "light" version. This is the point where I got directly involved in the implementation process, and I will continue this description in section 5.2.

4.5 The India case

DHIS 2 had in the autumn of 2006 gained a significant installed base in the state of Kerela in India, and another internationalization requirement was urgently requested. This case is directly related to my secondary research objective, which concerns the translations of persistent data. That the actual data in the database had to be translatable dynamically entailed a quite big change to the core of the system. To meet this requirement quickly an "alternative name" field was added to some of the objects, allowing objects to have two names. With the additions of an easily configurable sort-order option, it was possible to use these as translations and continue the implementation process. India has several official languages, such as Hindi, Gujarati, Telugu and Kannada (Wikipedia, 2007c), hence the system needed to be translatable into more than just one language to be accepted as a national standard. Additionally, more than just the name field needed to be translatable, making the need for a more flexible solution apparent. I developed a concept based on this case which is described in section 5.3.

5. Empirical study

My empirical study is based on my contribution to the DHIS 2 project from the spring of 2005 and throughout the spring semester of 2007. This period includes a field trip to Vietnam from the middle of February 2006 to the end of May 2006. During this time I had four distinct tasks:

- Continue the training of a local team.
- Continue collaboration with a local university.
- Implement overt internationalization capabilities for the user interface of DHIS 2.
- Start implementation of DHIS 2 in Ho Chi Minh City (HCMC).

Additionally, I present a generic concept for internationalization capabilities of data model objects. This implies a somewhat deeper approach to internationalization that goes beyond the user interface, and is based on the case from India (ref. 4.5). I conclude the development part of my empirical study by mentioning some thoughts about the DHIS 2 development and global collaboration. Lastly, the results form the survey introduced in section 3.3.1 is presented.

5.1 Internationalization of the user interface

The internationalization of DHIS 2 was completed in two steps. Although both of them deal with translations, the nature of these steps is very different and they will hence be described separately. This first step is described in the Vietnamese context in which it was developed, and it has a quite reductionist and instrumental perspective on localization with the overt requirements of enabling language translations and date and time formatting.

In addition to the team of 3 Vietnamese developers, I was joined by one Norwegian master student who was set to leave Vietnam in the middle of March. When the team had been established, we separated this process into three different tasks. Primarily we needed to add a level of abstraction to all strings printed in the system and replace all the usages with unique identifiers. Secondly, we needed a tool to perform these translations and define date and time formats. Lastly, we had to develop a method to align the resources with the rest of the system.

5.1.1 The i18n module

The process of enabling this feature was completed without any major issues. A general global resource file was created for very common expressions. In addition to this, each web module may define their own resource files, which are given precedence if both locations contain the same i18n key (ref. 2.7.3). The module was added to a predefined interceptor stack which made it available to the web templates.

In order to get the two-byte languages to display properly we had to set the web framework to enforce UTF encoding. As the HTTP protocol allows the client to set the supported encodings during the first get request this could be completed by configuring the web framework. This was hence solved by adding one line to a WebWork configuration file, enabling both UTF-8 encoding for data read from the resource bundles

and data input from the user. In the case of Vietnamese it was necessary to set this encoding type explicitly in the input software used.

The locale consisting of English as the language and the Great Britain as the location is the default locale of HISP. This was already established as the default locale in the HISP project when I joined the project, and I believe that the choice of this locale over its US alternative may be ideologically based. When keys are added, developers add translations for the default locale, and hence the component developed is immediately localized for the default locale. The i18n module will always fall back on translations from this locale for keys requested, hence strings may appear in English when a different locale is chosen and a translation doesn't exist for that particular key. The locale identification codes are suffixed after the module type name; hence the file i18n_global_vi_VN.properties will indicate that this global resource file contains Vietnamese translations for Vietnam.

Java has built in support patterns for both number and date formatting (Sun, 2007b) . This made enabling these overt factors a trivial process. Keys handling such patterns are all placed in the global resource file and prefixed by *format.* The formatting is then done on request by the i18n module based on the patterns specified. The same precedence rules applies, so if the format is not specified it will use the one defined in i18n_global_en_GB.properties. These standards were established quite quickly by first agreeing on the solutions within our local team and short discussions through the DHIS 2 developer mailing list (ref. 4.3.3).

5.1.2 The resource editor

After this was settled we had a clear view of how the resources should be defined, and we started surveying for editors to generate these resources. This did however not produce and satisfactory results, after a few days of testing various editors they all had limitations we could not live with. These limitations were either technical or problems with the license. As we needed to be able to distribute this as a package we required the license to be very liberal. After a discussion between the team members and the developer mailing list, the decision to create our own resource editor was reached. This would give us the

flexibility to align this product with the rest of DHIS 2, although it was not to be regarded as a part of the main DHIS 2 artifact.

While the others in the team started the lengthy task of extracting all sentences and phrases from the DHIS 2 user-interface, I created the first version of the DHIS 2 resource editor. A very basic version of the software was created within the first week to give the rest of the team the ability to start performing translations. During the next week the resource editor was improved based on the feedback from the team. All of the strings had been extracted from the GUI and English and Vietnamese resources were created. This proved to be a very efficient approach, though the quality of the code did suffer somewhat from the limited time we had on this.

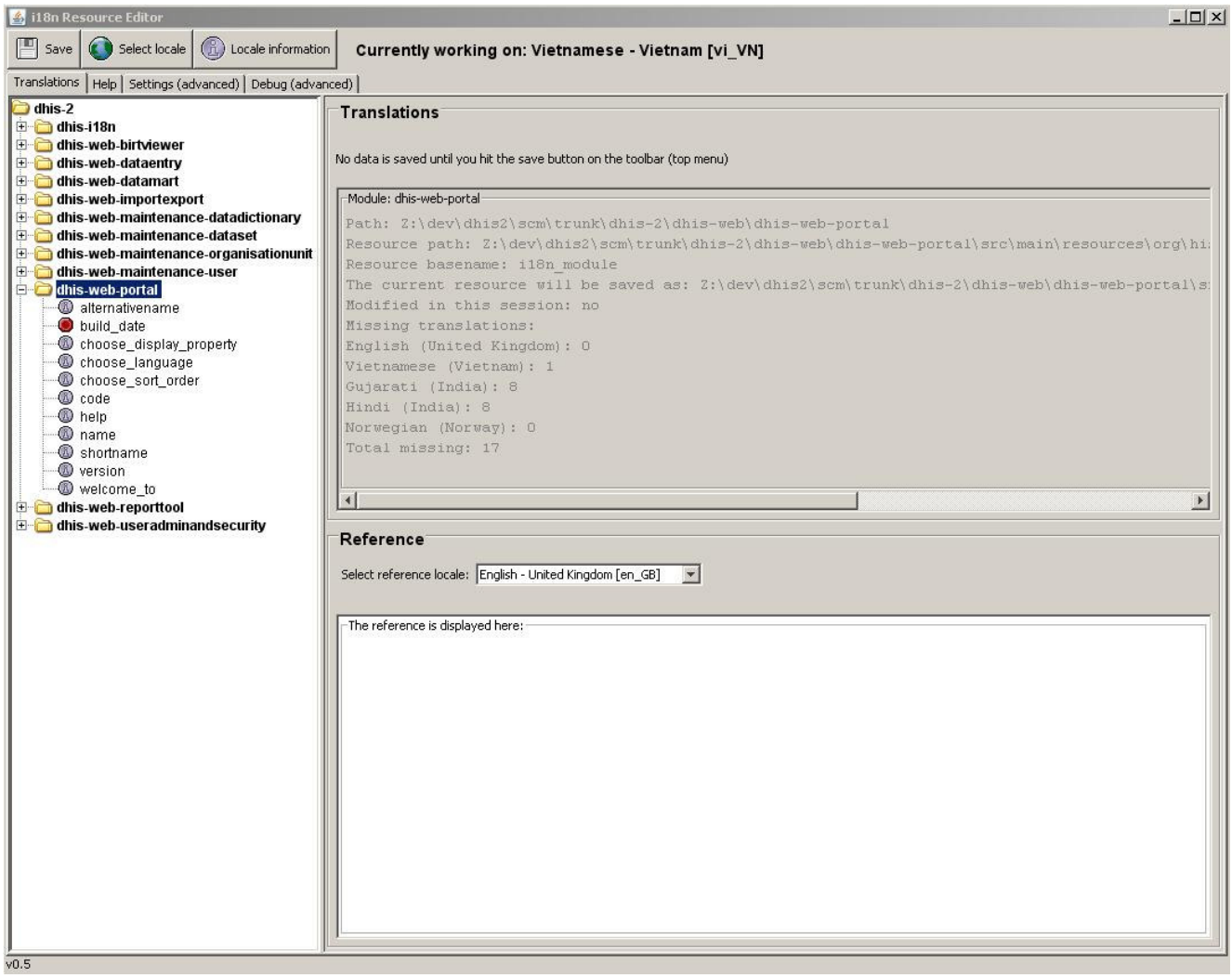

Figure 10: Screenshot of the i18n Resource Editor running in Windows XP

The support to edit multiple modules in one session and the summary of missing translations shown in the figure was added during a refactoring process in the spring of 2006 after my trip to Vietnam. When the system was localized for Vietnam only one module could be edited at the time. The rest of the user-interface is however left unchanged since the first version in March of 2006. My intention with this program has always been to write a general resource editing tool, usable by any project and not only DHIS 2. Currently the module tree shown in the figure is neither generic nor configurable enough to allow multi module editing, but its architecture doesn't exclude such flexibility to be added in the future. The look and feel of the application changes according to the

platform in which it is running on, and “special keys” like the date and time format patterns are shown in the italic font style in the right tree.

The resource editor also contains a validation engine that verifies the locale input from the user. The default list of locales supplied by the Java API both in terms of language and country codes proved to be insufficient in the case of DHIS 2. However, the Java API is able to lookup any value supplied, and gives a textual representation of that code. The validation engine takes advantage of this and tries to resolve any value, limiting the possibility of errors by the user. The first tab shows a list of locales which is a combination supplied by the Java API and existing resources. We decided to strongly recommend the use of both language and country code in the validation, but it will accept only the language code.

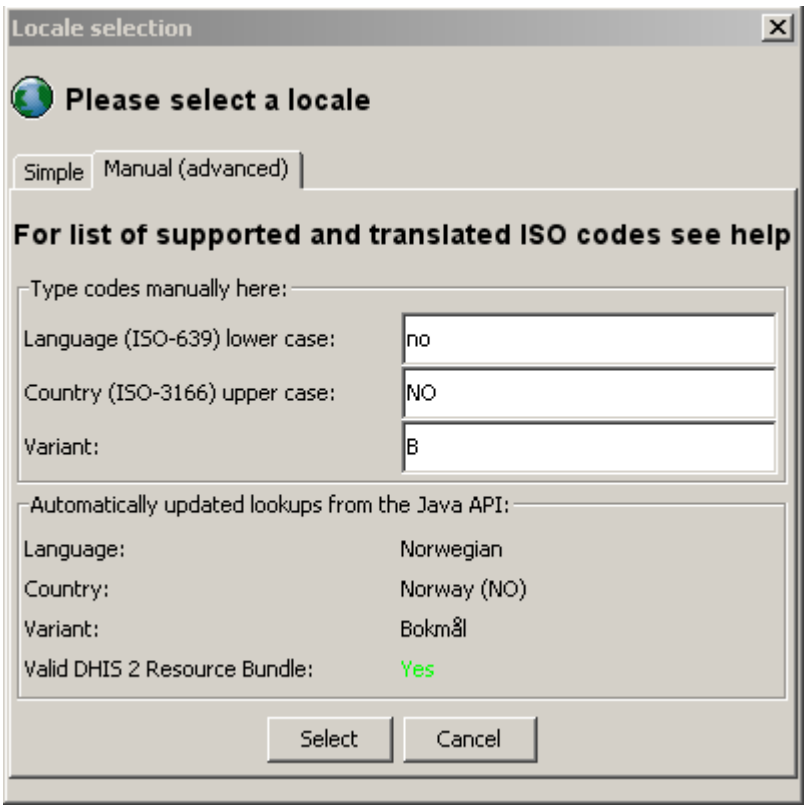

Figure 11: Locale validation

5.1.3 Submitting translations

For the task of submitting resources to the system, we chose to use the existing infrastructures provided by the version control services of the source code repository which is provided by Subversion (ref. 4.3.3). Figure 12 demonstrates the key transmission points in the translation process:

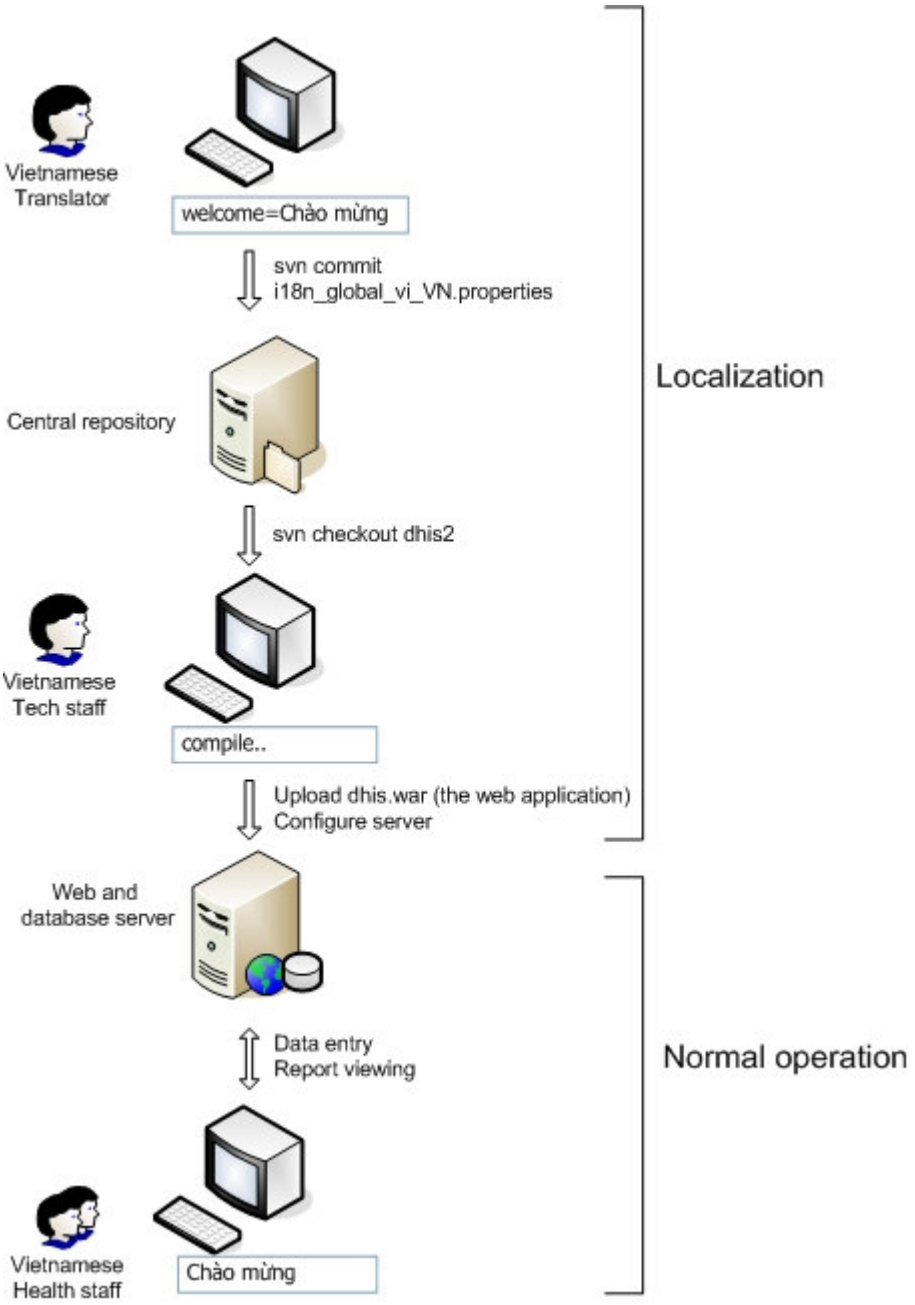

Figure 12: DHIS 2 i18n translation information flow

First date and time formatting are defined and resources translated using the i18n resource editor. Then the translated resources are committed to a central repository. The second and third transition is the checkout and compilation process done by the tech staff responsible for the deployment. The fourth transition describes the normal operation of the applications, as health workers are entering and retrieving information from the system (getting their welcome message). Notice how localization covers both the translations and the preparation for deployment displayed and that Vietnamese staff is able to handle all localization processes locally.

This figure is created to indicate the main actors and the general internationalization information flow. It does not include any testing requirements or consider the translator's need to know the context in which the strings are used. In many cases the role of the translator and the tech staff may be the same person or team. The web and database server and the health staff computer are also often the same machine in real world deployments.

This approach to translations is something that most local teams have been comfortable with, which is reflected by the list of language resources available with the Milestone 7 release in April 2007:

- Amharic (Ethiopia)
- English (United Kingdom)
- French (France)
- Gujarati (India)
- Hindi (India)
- Kannanda (India)
- Malayalam (India)
- Norwegian (Norway)
- Telugu (India)
- Vietnamese (Vietnam)

5.2 HCMC localization

The internationalization of the user interface was just one enabling factor in the localization process. This section explains how DHIS 2 was translated and implemented in the Vietnamese context. It is not a study of the implementation process of DHIS in Vietnam, but rather a study of the localization capabilities of DHIS 2. I will adopt terms from the information transfer life cycle of Baark and Heeks (Figure 1) to position the localization in the larger scope of the implementation process. The choice of technology had not been taken prior to my arrival to Vietnam, and the choice of implementing DHIS 1.4 was still on the table at the time. A successful pilot project had been completed in the state of Kerela in India at the time, and we felt that DHIS 2 was ready for deployment in Vietnam. The addition of internationalization was also a contributing factor to the choice of implementing DHIS 2. Hence, the assimilation and adaption of DHIS 2 was the main focus of my field study from that point. We had during the development of internationalization capabilities invested a significant amount of time into the assimilation of the Vietnamese team to the DHIS 2 technology, their capabilities in contributing to a sustainable solution through regular maintenance was of great importance – especially due to the troubled history of DHIS in Vietnam (ref. 4.4.8).

5.2.1 Demographics and infrastructure

The essential demographics of HCMC in this context is that the total population is 6 245 0000 in 2005 (HCMCPC, 2007) and that the total amount of top level organisational units are 26. This number includes 7 hospitals and the rest are districts. All locations are quite easily accessible public transport so distance did not pose any major challenges for us. My main priority during the pilot phase was to get implementation running at least one district and one top level hospital before I left. The datasets of these two types were a bit different, so it was important to get them both tested and approved locally.

5.2.2 Preparation

The history of DHIS in Vietnam had led to frustration from the health staff and my impression was that it as a very low level of excitement from the workers. This made it critical for the system to work seamlessly from the beginning, and a lot of preparation

was put into the first pilot implementations. After the DHIS 2 application had been internationalized it met the requirements for implementation in Vietnam. We were introduced to a hospital of Mother & Child health in HCMC where we also rented an office. The approach of implementing DHIS 2 directly was appreciated by the staff at the Mother & Child health center, as they thought that this would be better received by the end users. Implementing 1.4 would merely be regarded as an upgrade of an existing system that did not work properly in the first place, whereas DHIS 2 would be seen as a replacement.

5.2.3 Roles

I will briefly explain the main actors in the localization process. When the implementation started our team was as, as planned, reduced by one developer, as the other Norwegian master student went home. This left us with three Vietnamese developers hired by HISP and me as a team leader. We had a contact the Mother & Child health center who was our primary communication with the health community. He also acted as the network administrator at the facility and he had knowledge about the computer hardware at the health facilities. This made him an important part of the establishment of technical requirements for the localization of DHIS 2. Additionally, there was a medical doctor that acts as the main HISP contact in the country in terms of payments and logistics. When the implementation started we were joined by a Vietnamese master student formerly employed by OutSoft, who had been a central actor in the development of DHIS 1.3. He assisted us in the establishment of contacts at the local hospitals and health stations that helped us to get implementation approval to move forward.

5.2.4 Implementation plan

A document that established the basic relationship between the main actors and the scope of the initial deployment was needed. In cooperation with the coordinators in Oslo I created the first draft of this plan (Appendix B), which was circulated to the main actors in the end of March. Some actors felt that the progress was to slow, and the plan was opposed by actors within our team and in the hospital health staff. Experience from recent implementations of DHIS 2 thought us that this progress was the safest approach,

especially in light of the opposition the system had at the district level we could not afford any serious mistakes. With support from Oslo we eventually managed to align the interests of all actors involved, and implementation was set to start in the beginning of May. At the meeting where the plan was approved, our contact at the Mother & Child health care center was given extended authority from the leaders at the center to approve the software ready for implementation.

5.2.5 Installation package and technology choices

DHIS 2 is as mentioned highly configurable in terms of supported services and local configuration. It did not have any standard installation procedure at the time, and it became clear to us that this was needed for HCMC. Working with the staff at the Mother & Child health center, we came up with a set of requirements and requests that would be best performed by an automatic procedure at the districts. We had only computers running Windows XP and Windows 2000 to consider in this context, and the following requirements were presented:

- DHIS 2 services should not run as the administrator user.
- The installation procedure should be fully automatic.
- It has to support low-resolution monitors (800x600 pixels).
- A single shortcut on the Desktop should start DHIS 2 and all its services, making DHIS 2 appear like its predecessor.

First we had to find an installation framework with a license that allowed us to freely modify and distribute it. After a day of researching alternatives we ended up with a package called “Installer2GO” (SDS, 2007), which provides the ability of executing batch scripts. Most of the requirements presented could be met quickly by writing Windows batch scripts. The requirement to meet the 800x600 pixel resolution required tweaks to the GUI, as the lowest supported resolution of DHIS 2 at that time was 1024x768 pixels. This was however achieved by modifying a style sheet (W3, 2007b) that was located in one web module, and did not pose any major challenges.

A considerable effort had to be put into upgrading the database from the former implementations of DHIS 1.3. This was not a trivial process as the database structure is somewhat different. We did manage to recover all the data into DHIS 2 and we got the hierarchy of the organization units based (ref. 4.2.1) on this process. All of the recovered data was bundled with the installer of DHIS 2 to the local health facilities. It is however a time gap of about a year between when the health facilities stopped using DHIS 1.3 and the DHIS 2 implementations. We were informed that staff at the health center would register these values manually based on paper reports.

As a part of the installation package we also included a highly graphical Vietnamese manual for DHIS 2 and an introductory video, which unfortunately became particularized for Vietnam and never translated to English. We planned to do the basic training on site during the pilot phase, and hold courses at the medical university for the full scale deployment.

5.2.6 Implementation

The Vietnamese team hired by HISP at the beginning reluctant to perform implementation tasks. Up until then we had worked on the static internationalization solution and other programming tasks, and the team was eager to continue with development. The educational element of the job was a central part of their motivation to work with HISP, and they could have made more money elsewhere. A lack of support was as mentioned one of the main reasons for the 1.3 implementation to fail (ref. 4.4.8). However, this time the staff was directly hired by HISP and not dependent on a third party such as OutSoft, and after some persuasion from me personally we decided to do the implementation. Assurances were given that they would have development tasks after the implementation was done in addition to supporting the solution in HCMC. In practice, minor development tasks also surfaced while getting requirements during the localization, which the team was able to handle very well. This included a graphical date selection script that was committed to the central DHIS 2 repository and various other local changes like tweaks to the GUI and configuration files. Some innovation emerged

from this process, as the date selection script was accepted as part of the main artifact of DHIS 2 and has been a part of all releases since.

The implementation in HCMC proceeded mostly according to plan. The installation package worked without a hitch, and we got seemingly good reviews from the health staff. By seemingly I refer to this by facial expressions, body language and the translations I got from the team. The English proficiency proved to be very low at the districts, but the localized user-manual and video was well received. As part of the first implementation we did linguistic tests and corrections, and 3 iterations were needed in order to get the medical terms and reports right. Other tweaks to the application were also performed during this period. Contact was established through the exchange of phone numbers and emails, and most locations implemented during the pilot phase had to be supported several times after implementation. This was however a task that we were able to perform without much trouble, as we now had a team that was actually devoted to do this.

At the time we had a visitor from the Hue province who was there to observe DHIS 2. This was also very valuable for us, as she was the only one in the team with any real experience in implementation of DHIS (DHIS 1.3). Later she benefited from this experience when she along with three other master students from the University of Oslo successfully implemented DHIS 2 in the Hue province in the fall of 2006 (Øverland, 2006).

This concludes my direct involvement in the implementation effort. Further implementations went according to the plan outlined in the implementation plan and by august the whole city of HCMC was covered (Øverland, 2006). Sufficient trust seemed to be built at that point, and no official documents were needed to cover the whole city. During the preparation for implementation we were given the opportunity to use an auditorium at the medical university for user training, which was carried out successfully in august 2006.

5.2.7 Online deployment

DHIS 2 is web based and it can hence be deployed as a centralized solution serving multiple clients (ref. 4.2.3). While waiting for final approval to start the implementation from the authorities, I started to look at the possibility of implementing DHIS 2 online. The infrastructure in Vietnam does not allow such deployment at all sites, but where available it would help to decrease the complexity of the implementation process. Not only in terms of maintenance, but it would also to some degree relieve the system of import and export routines between the health facilities.

A Norwegian developer had knowledge of open source framework that could handle online authorization, and along with a Vietnamese developer in our team he developed a very basic login system quickly. This was purely a matter of prioritizing as such functionality exists in DHIS 1.4, and it was scheduled for a later release of DHIS 2. A Vietnam specific demo version of the system was set up on a HISP server in the UK, and possible technical problems with response times were ruled out. After an initial skepticism, I was able to get approval for a pilot project of this solution from the HISP Vietnam contact and the director of the Mother & Child health center.

The online initiative had to shut down after talking to the managers at the local health stations. The reason was that they would not agree to give their employee's access to the Internet. Arguments of the practicality of such a solution all feel through. The cost of labor in Vietnam is very low and hence not seen as an issue by the local management. Giving the employees the freedom to use the internet was regarded as a much greater risk to take. Even where available at the health facilities, I observed that internet was disconnected from computers used by parts of the health staff.

However, as the system gains more trust from the government this situation may change, and the use of online versions may be enforced by higher authorities. The team has since the first deployment kept track of the possibility of online access at the sites. In addition to this the hardware at the deployment sites has been kept. This may prove useful in the

future to determine if new features added to DHIS 2 can be run efficiently by its end users.

5.3 Internationalization of the persistence layer

In this section I will describe how persistent data can be made translatable in an existing software project. This concept is rather technical in nature, but it does not necessarily rely on a specific programming language. I will illustrate this concept with a case study on how this was applied in DHIS 2. Its development was less of a team effort than the first internationalization solution, as the programming was done by me personally. Other developers were however of great help, and all major decisions were taken by consensus on the developer mailing list. I have not done any assessment as to the scope of the concept presented here in terms of where it can be applied.

5.3.1 Motivation

The motivation for such as solution was in the case of DHIS 2 to a large degree based on the need to avoid particularization of the system for India. We had already seen movement towards this with the addition of the alternative name field, and they requested this to be added for several other data model objects. By aligning this solution with the rest of DHIS 2, we translated the requirement from India into a solution that could potentially be used by all users. It is also something that is likely to be a condition for DHIS 2 in order to scale up to being accepted as a national standard in countries with multiple official languages.

This functionality would also help the global development team to support local solutions. From my own experience gained while working on this solution I know that it is often difficult just to tell the difference between objects in localized databases, especially when they are named using complex script languages such as Hindi. An enabling of translations on this layer would allow the Indian teams to translate at least the main objects in question before requesting community support.

5.3.2 Requirements

Based on discussions on the developer mailing list, a set of requirements was reached:

A. Existing installed bases need to be upgradeable, including an optional conversion of the alternative name field into proper translations.
B. The system was not designed to support this, making changes to the system nearly inevitable. There were however a wish from the community to limit these implications.
C. A wish to keep the original "POJO" object structure (ref. 4.3.1).
D. To be regarded as an optional feature, hence it should not "force" the users to use this feature and align well with the rest of the system.
E. Users had to be able to use the database directly, either by other software or direct queries. Hence the database had to contain data for one locale consistently.
F. Between 1 and 4 fields of 9 different object types needed to be translatable.
G. A user-interface to perform translations should exist with the ability to lookup existing translations as reference.

5.3.3 Implications

When I first started looking at this I was quite surprised as to exactly how much this would impact the whole system. Translations had to be considered in most aspects of the system, but first one key decision had to be made.

It became clear that one locale (herby referred to as the "fallback locale") had to be defined. This would be the locale that data in existing databases would be bound to during the upgrade process, and the locale for the data in the database at all times (requirement E). An experienced developer of DHIS 1.x advised me to not use the same locale for this and the user-interface solution, as the implementations of DHIS 1.4 in South-Africa showed that users often want to use this functionality independently. Separating these configurations entirely made sense also in regard to the design of DHIS 2, and the decision to go with this approach was taken quickly.

In the context of object translations it is convenient to reduce DHIS 2 to an input and output system where data is collected and presented. These operations are commonly

referred to by the term CRUD operations (Wikipedia, 2007b), which is short for Create, Read, Update and Delete:

- Create: create or add new entries
- Read: retrieve or view existing entries
- Update: update or edit existing entries
- Delete: delete existing entries

DHIS 2 has very basic standard dialogs for all of these operations, and there was a wish to keep them as they were. These four basic operations presented requirements that had to be dealt with in separate ways:

Create: This was the operation causing the biggest challenge. Basically we had two options here; either enforce creating an object using the fallback locale or accept objects in the database with a different language. After a discussion we decided to use the last option as the first option wouldn't be possible to enforce in all implementation scenarios. Once an object is updated using the fallback locale, this data will get saved to the database. Hence, when an object is created using a different locale than the fallback locale, data is kept in a translation table so nothing is lost when it's overwritten with data from the fallback locale.

Read: When an object is read all fields enabled for translations need to be updated according to the locale currently in use.

Update: Update operations are quite similar to the create operation. Internationalized data need to be saved to translation tables, and the object fields need to be updated with data from the fallback locale before saved to the database.

Delete: When an object is deleted from the system, all translations for that object should also be removed for all locales.

5.3.4 Identify objects

As mentioned 9 objects was identified to be internationalized. This amount of objects clearly supported a generic approach. To satisfy the requirement of alignment with the rest of the system (requirement D), it would also be useful to have a generic solution for the generation of a user interface for translations (requirement G). We also wanted this solution to align with the overall software design, and the definition of these objects should be separated from the source code. I will demonstrate how this was implemented by the use of two model objects, DataElement and DataElementGroup:

```
DataElement
id : int
uuid : String
name : String  (i18n)
alternativeName : String
shortName : String (i18n)
code : String
description : String (i18n)
active : boolean
type : String
aggregationOperator : String
parent : DataElement
children : Set<DataElement>
```

Object description 1: DataElement

```
DataElementGroup
id : int
uuid : String
name : String (i18n)
members : Set<DataElement>
```

Object description 2: DataElementGroup

As we can see from the object descriptions, there is a many to one relationship between these two objects. Additionally, the object model allows DataElements to be organized in hierarchies, but this is not in use yet in DHIS 2. I have suffixed properties that should be i18n enabled with i18n in parenthesis. The need for a generic solution led me to investigate how the Spring framework could be used for this. As mentioned Spring uses Dependency Injection to set properties in classes (ref. 4.3.2). These classes are

implementations of interfaces that do not necessarily have methods to set these properties defined. The technique used to call the methods is usually referred to as reflection, and it's commonly available in most major high-level programming languages. It allows execution of methods on objects without knowing that they exist. The type of the class is always available in Java as a String value, hence this can be matched against a predefined definition. With only POJO objects to consider, this was all the information I needed to define a generic i18n solution.

Using Spring for the definition and reflection for setting the properties, I came up with the following solution. At the core of the definition is a very basic I18nObject containing a list of all properties that should be internationalized for that particular object. Instances for this object are defined for each object type that should be internationalization enabled, and fieldnames are supplied by the use of a list property. In addition to this definition, the name of the class needs to be defined. In the case of DHIS 2 we decided to only use the name of the class here, but the whole path including package can of course also in most cases be used. This definition can be realized in many different ways using other frameworks, but I would like to emphasize how keeping it away from the programming source code increases readability. The key requirement behind the name definition is just that it needs to be able to retrieve the name of a class without knowing its object type; in Java, all objects are able to return their own class object, which can return a textual representation of its path.

The configuration for the object looks like this in DHIS 2:

```
<bean id="I18nDataElement"
      class="org.hisp.dhis.i18n.db.I18nObject">
  <property name="className" value="DataElement"/>
  <property name="propertyNames">
    <list>
       <value>name</value>
       <value>shortName</value>
       <value>description</value>
    </list>
  </property>
</bean>
```

```
<bean id="I18nDataElementGroup"
      class="org.hisp.dhis.i18n.I18nObject">
  <property name="className" value="DataElementGroup"/>
  <property name="propertyNames">
    <list>
       <value>name</value>
    </list>
  </property>
</bean>
```

Code example 1: Internationalization definition for two objects configured through Spring

The i18n object beans are then registered at the i18n service:

```
<bean id="org.hisp.dhis.i18n.I18nService"
      class="org.hisp.dhis.i18n.db.DefaultI18nService">
  <property name="objects">
    <list>
      <ref bean="I18nDataElement"/>
      <ref bean="I18nDataElementGroup"/>
      <-- cut -->
     </list>
  </property>
</bean>
```

Code example 2: Registration of internationalization definitions

This is all the i18n service needs in order to enable these objects for internationalization.

5.3.5 Persistence

A number of methods that can be used for persistence in this concept, and I will not go into details about alternatives. In DHIS 2 we decided to use composite keys, but stayed with an OR approach. An object would contain only one specific translation for one field of one particular object. This would increase the readability of the database schema, which was important for DHIS 2 users using the database directly.

```
Translation
className : String
id : int
locale : String
property : String
value : String
```

Object description 3: The Translation object

The first four properties in code example 3 constitute the composite key, and a Translation store module was created to handle the persistence of this object type. Some projects have issues with the use of composite keys and it's of course possible to just add another field and avoid using it, but the composite key approach is a convenient way of avoiding duplicates. The use of the I18nService makes how the persistence is organized transparent, and developers in need of internationalization for object will only have to deal the service layer definition described above. This persistence can be handled more efficiently and in a more object oriented manner without the requirement of human readability. With this approach a few exceptions had to be made that may be at odds with the ORM methodology, such a query to get all available locales without retrieving all Translation objects. This was an acceptable solution in the case of DHIS 2, as I used HQL (ref. 4.3.2) for this query causing the database abstraction added by Hibernate to be intact.

5.3.6 Implementation

Although this is an aspect oriented approach, it can be implemented by many different ways. By implementation, I refer to the points where the i18n service actually is manipulating the objects. This will of course depend on the frameworks and technology available, I have chosen to group implementation alternatives into three distinct groups:

Event based: The use of event implies that events are triggered when needed instead of explicitly called. By using events for the internationalization it can be completely transparent to the other code. Adding such transparency can arguably be dangerous, as objects get manipulated without explicit calls where developers normally would expect them. Using such a model would also imply that all objects would get checked, and not only the i18n enabled objects. This is however a very generic approach that would require a very low amount of code.

Direct pragmatic interception: A more obvious option is to intercept the calls in need of internationalization directly at some point in the system. In the case of three-tier architectures this would be likely to be done either on the service or persistence tier. By

implementing internationalization using this approach it's possible to explicitly control all the usages. It has a clear advantage in that it has less overhead than the event based model, as only the i18n enabled objects will be checked.

Aspect Oriented: Using an AOP framework, such as the one provided by Spring, it's possible to fully implement i18n as an aspect of the system. This implementation will be similar to a common way of implementing transaction management, and will have to be aligned with such solutions. Such a solution will to a large degree provide the transparency provided by an event based solution, but without the overhead caused by all objects being checked.

Based on how object relations are loaded with Hibernate (ref. 4.3.2) it was necessary in the case of DHIS 2 to use a data-tier event handler for the reading of objects, as those calls often never passes through the service layer. Tests showed that this issue did not have any significant overhead in the case of DHIS 2, but this may turn out differently in other projects. We encountered a conflict between how persistence sessions are managed that delayed a fully AOP based implementation. Hence, we decided to go for a safe solution of intercepting all calls related to the remaining three CRUD operations pragmatically. It was hence wired by Spring and implemented pragmatically as an aspect of the system at the service layer.

5.3.7 GUI

A generic approach can also be used to create a user-interface solution for the translations. I will illustrate briefly how we aligned object internationalization with the DHIS 2 user interface. The principle of this approach should work for most projects, even if they are not web-based. To align i18n with the CRUD operations of the system, I added translations after the edit option in the operations column.

Figure 13: I18n GUI alignment

By making the definitions from code example 1 available through the i18n service, I was able to create a general translation GUI system for DHIS 2 that is able to generate translation forms for all i18n enabled objects. Based on the requirements presented a basic GUI was created. Asynchronous JavaScript and XML (AJAX) requests are used to look up reference translations, making them appear directly on request. A form was also added for adding new locales to the system.

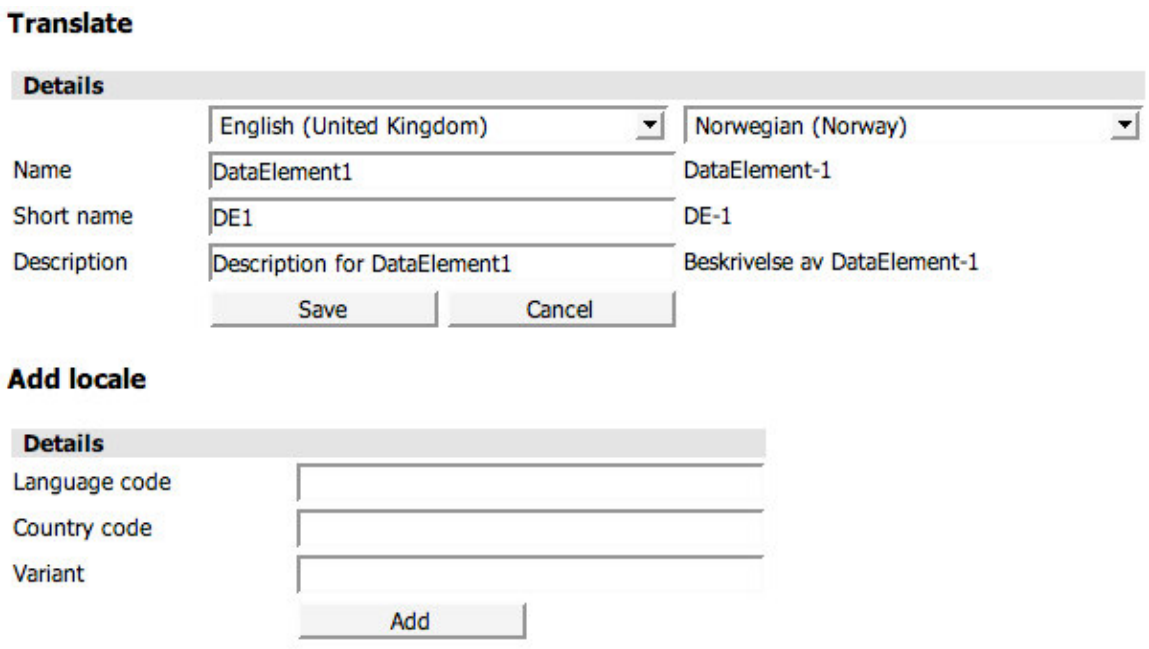

Figure 14: Translation GUI

5.3.8 Upgrade

Because of the exiting installed base we need a reliable and stable upgrade tool. I wanted to take advantage of the database abstraction provided Hibernate for this, as it made it possible to have a general solution not depending on any particular database server. Keeping in mind that not all locations would need this feature, it should not be too intrusive on the users. It was however necessary to define the fallback locale, as all the existing data should be mapped to this locale. This locale definition would default to the default DHIS 2 locale, which is English and Great Britain. Hence users not configuring this would have all their data bound to this locale. It was also a wish to be able to transfer

the alternative name property used in India, so this was included as an optional feature disabled by default. This procedure simply moved the alternative name property to the name property for each object for a configurable second locale.

We decided to let the upgrade procedure run automatically during the startup procedure of DHIS 2. Some overhead was expected by the use of Hibernate instead of native SQL, but when testing this with the largest available database the upgrade procedure never exceeded 3 minutes on any of the computers it was tested on in Oslo. It will take longer on slower systems, but the upgrade procedure will only run if the database doesn't contain any translations. Hence, we did not regard this as an issue as the upgrade procedure in most cases only would run once.

5.4 DHIS 2 development

The technical perspective of my empirical base is mainly concerned with internationalization and localization capabilities of the system. There are however other aspects of the system that is relevant to localization and my research, but do not involve the modules described so far directly and these will be mentioned in this section.

5.4.1 The import/export module

Localization deals with all translation of software and can as mentioned go beyond the scope of features made available through internationalization features. One key example of this from the DHIS 2 project is the development of the import/export module, which deals with importing and exporting between instances of DHIS 2 and exports to other formats. I have been involved with the development of this module, mainly on the user-interface side and the export of calculated indicators to other systems. I have never been the main architect of this module, and its development will not be described in detail. I will instead use this as an example on how requirements can be managed in OSS projects.

Based on a requirement from India, a relatively simple functionality to support the export of comma separated files to a fixed format was developed. After a workshop in South-Africa where most development nodes were represented, it changed into a more general

solution. After aligning the interests of the users and the developers, the fixed format was replaced by a plug-in architecture where all users were able to develop export formats freely and add them to the module. This is an example on how one idea based on a local specific requirement can materialize and lead to innovation that can benefit the whole development network.

5.4.2 Global collaboration

The development of the user-interface internationalization, the Vietnamese localization and the implementation was done in collaboration with the Vietnamese team. Additionally, I was involved in collaboration with a partner university outside HCMC, where a student group consisting of 8 students was developing a module for DHIS 2 as a school project. I will in this section present some thoughts around the collaboration in this context.

The most difficult challenge was definitely to include the locals in the online communication. The hired team had worked with two fellow Norwegian master students prior to my arrival, but they were yet not comfortable with the online communication. The use of the developer mailing list is the corner stone of the communication in DHIS 2, and a key purpose of our field trips to Vietnam was to establish such teams. Strong persuasion over a period of several months proved to be necessary in order to establish such communication practices.

I made a conscious effort of "making" a Vietnamese team member report bugs when they were discovered. An example of this emerged from the localization effort of DHIS 2 for HCMC. While working with the latest release of DHIS 2 we discovered a violation of SQL standards, which was picked up by the PostgreSQL server. Namely use of the reserved keyword "user" in the creation of users for the secure login functionality. We were the only location using this particular SQL server, and this information was useful for the project globally. After some persuasion a Vietnamese team member sent a bug report to the developer mailing list and it was recognized and fixed by a core developer in Oslo, who was responsible for this particular module, within the same day. This approach

was repeated several times, and by the time I left they were using the developers' mailing list by their own initiative. Such communication did however not materialize at the partner university. Interviews suggest that the primary reason for this is the time spent there, which in my case was just 6 personal meetings with the group.

<Question of reasons for the failure of establishment of communications with the partner university>

(Vietnamese developer 1) first is the language problem, second is they didn't work with us closely, so they were afraid of it, and they don't know who will answer, and how are they, so ... (Vietnamese developer)

(me): I remember that it was a little bit difficult to get <mentioning names of team members> to use the developers list at first too, the same reason?

(Vietnamese developer 1) yes, I think but the biggest problem is the language, I think sometimes, I know the problem but cannot express it in English and sometimes, when got your reply, cannot understand. I used to spend all day to understand one email :(

(Vietnamese developer 1) ah, about the reason for not using the dev list at the beginning we was not familiar with doing that, and still shy to do that and we didn't really understand the system

(Vietnamese developer 2) like us before
(Vietnamese developer 2) cannot speak and listenning
(Vietnamese developer 2) just can read, with help from dictionary
(me) yeah.. need to get over that :)
(Vietnamese developer 2) :)
(me) difficult for us to tell them to speak.. like with the <university name> network group
(Vietnamese developer 2) yes

Chat transcript 1: Vietnamese developers about online collaboration

The last statement by Vietnamese developer 1 is particularly interesting, as the interview subject points to the communication forum as a reason. A quite casual tone is used in the developer mailing list, and I can understand how this may be difficult to pick up on. The responses strongly suggest that the collaboration with the university would have been more successful if more time was spent there. The collaboration was picked up again by a Norwegian master student in August of 2007, but the implementation of DHIS 2 in Hue was of priority at the time and this process also died out.

We did, as explained, undertake several programming tasks during the course of my empirical study. Also in these cases it proved to be necessary to teach some aspects of the development, such as principles of good programming practices and modularization. The Vietnamese team did however according to my observations generally have good general programming knowledge, and they had very little difficulties in coping with the complexities of the frameworks in DHIS 2. I would also like to point out that they both in this case and in the case of online communication showed a willingness to learn and seemed interested in global development.

Upon completion of tasks, there was however according to my observations a certain lack of initiative from the Vietnamese team. My impression was that the primary reason for this is that they are used to being "told what to do" by some higher authority. An observation done by Øverland (2006) at a university in Vietnam confirms the same tendencies:

"The students were disciplined, motivated and skilful; they spent approximately eight ours at school every day, worked hard with their assigned tasks, and had obtained a considerable base of knowledge for DHIS 2 development (ref. 9.6.3). On the other hand they suffered from a deficit of individual initiative and the ability for independent problem solving. In order to achieve progress they had to be given specific work tasks. Incidents from the first phase of my training and co-development with the students where the students started playing music and talking about everyday matters after finishing assignments (ref. 9.6.1), emphasise this assertion. The students' attitude was apparently partly a consequence of the education scheme of the Vietnamese educational system." (Øverland, 2006, p. 108)

The same strict guidelines were present in the studies of Nordal (2006) at a software company. According to my observations no real contribution in terms of direct development of the main artifact of DHIS 2 has been done during my time in the

development process at any of the development nodes in the developing countries without the direct participation of western personnel.

A disagreement over working contracts recently lead the Vietnamese HISP team to leave the project and pursue a career working for commercial Vietnamese software companies. Currently a new team has been established by the former OutSoft employee, but they are not proficient in English and not visible in the online development network of DHIS 2.

5.4.3 Local specific development

Systems have however emerged locally, and one of them (HISPIndia, 2007) is based on resources generated by the DHIS 2 application directly to the database instead of through the service layer (API) of DHIS 2. This system is the primary reason for requirement E of the translations of persistent data (ref. 5.3.2), and it is a source of tension between the developers of DHIS 2 and developers doing localization work. Using the service layer is something that is strongly encouraged by the programmers globally, as it abstracts changes to the underlying persistence. When database changes are introduced, either by the introduction of new features or refactoring, users using the database directly are often required to update the database manually. After a refactoring that introduced database changes to the system, an Indian developer raised this question on the developer mailing list:

“Yet another database changes, are users going to use database or DHIS 2.0? (Indian developer)”

Whereby a core developer replied (in relation to a following discussion about users who use the database directly):

“Yes, this was a major concern from day one. If people start "integrating" with DHIS by going directly to the database - it will likely at some point grind the development of DHIS 2 to a halt, as it will become practically impossible to do changes in the DHIS core. (DHIS 2 core developer)”

This highlights a source of conflict that according my observations has intensified as DHIS 2 has evolved.

5.4.4 Issue tracker

The use of the issue tracker (JIRA) (ref. 4.3.3) faded out during the fall of 2006. As no developers in developing countries filed issues, the use of such a tracker became an isolated and tedious process. This tendency was picked up by Nordal (2006) in the second quarter of that year, and a solution was proposed:

"Experiences from the DHIS 2 project suggest that a simplified way of registering issues should be available for the users. Either by letting the users send their issues to the mailing list, and have a person responsible for registering those issues in the tracker." (Nordal, 2006, p. 97)

In May 2007 the communication tool Trac (Edgewall.org, 2007) was introduced that simplified this process as requested by Nordal (2006). It's a basic issue tracker, which is less technically advanced than JIRA, but it has been very well received by the DHIS 2 development community. Although this is less technically sophisticated than JIRA, it incorporates the features currently needed in DHIS 2 development. Additionally, it integrates with the mailing list and provides its own Wiki solution. The Wiki solution has, as of July 2007, not replaced Confluence, but I know that this is under consideration for the DHIS 2 specific parts of the HISP Wiki. Trac provides a user-friendly way of submitting issues (named "tickets"), and it integrates with the developers mailing list. It will have to be used for a longer period of time to be properly evaluated, but my observations of the use of this tool over approximately two months is that it looks very promising. It was almost immediately adopted by nodes in developing countries, and emails automatically sent from the tracker to the developer's mailing list when issues are filed are used as base for further discussions.

5.5 Survey

I will in this section present the results from the survey that was presented in section 3.2. This survey was partly done to verify some of my own assumptions about DHIS 2 and localization, but I also hoped to uncover other aspects of DHIS 2`s localization abilities. I did as mentioned end up using a qualitative approach, where I combined the survey with a series of interviews to clarify positions taken in the survey. The survey is based on answers from 7 individuals who were situated in Vietnam or India in May 2007. While this sampling is low, the respondents play key roles in the areas surveyed and I do think that they represent the opinions for the teams working in those areas, but I have no way of verifying this.

5.5.1 Results

I will start by presenting the numeric results. This represents the average score of the survey in terms of respondents and it is not evenly distributed among the locations. Respondents were asked to rate the statements truthfulness on a scale from 1-6, for the complete survey see Appendix C.

Part 1: User-interface translations

1.1 The DHIS 2 resource editor (the translator program) is easy to use: 5

1.2 The process of submitting translations to the project is straight forward: 5.5

1.3 I have not experienced input related and/or encoding related issues while translating (Text appeared correctly in DHIS 2 and/or the translation program after translation): 5

Part 2: DHIS 2 localization

2.1 DHIS 2 is easy to set up and deploy locally: 3.67

2.2 The answers from the DHIS 2/HISP community are informative and helpful: 4.17

2.3 The response time from the DHIS 2 community is adequate: 5.33

2.4 DHIS 2 adapts well to local context(s) and users find it easy to use: 4.17

2.5 Local requests made to the DHIS 2 community are being heard: 4.33

2.6 DHIS 2 is working well on the computers where it is deployed (in terms of speed and other hardware requirements): 3.83

Part 3: Database Internationalization

3.1 Its easy to translate objects in the database: 4.5

3.2 When upgrading existing installations to support database internationalization no problems were encountered: 6

It's necessary to investigate the reason behind these statements further, and I will now do so by giving a summary of comments from the survey and the interviews. Some respondents provided more comments than others, and not all respondents have been interviewed. Interviews were primarily done to get elaborations on answers from the survey and some clarifications were needed in order to fully understand the answers.

Starting with the translations of the user-interface, it seems like its well received by all users surveyed. Further, my survey points to an emerging tension between the developing node in Oslo and nodes in the developing countries. All of the respondents that have dealt with implementations point to a lack of response from the DHIS 2 developers when new functions are requested. A lack of decision making is commonly highlighted as a major frustration here, as discussions often "die out" without any decisions being reached. An interesting observation is that the response time is acceptable, but the answers given often not considered sufficient. Many regarded this as a "blocker" in terms of deployment, and several respondents expressed concerns that developers may be more interested in discussing the technology than creating actual features.

Users generally perceive DHIS 2 as an unnecessarily complicated product to deal with in terms of deployment. As it is very rarely used as a centralized web application, but rather as a desktop application that happens to use web technology a user points out how complicated it is compared to its predecessors. The platform independence and flexibility provided by DHIS 2 don't seem to be of importance to the users, all participants asked used DHIS 2 locally on Windows based computers. In India they identified the process of

actually deploying DHIS 2 locally as the single most challenging part of the localization process, because of the manual labor involved in each case. Most respondents expressed that they have problems following the online discussions, both linguistically and technically.

Performance is also an issue, and there is a common concern that the gap between the computers used to develop DHIS 2 and the ones used by the end users are too big. Especially in terms of memory and CPU power available to DHIS 2, but also the need to support lower screen resolutions is emphasized. The fact that it's a web application seem to be a disadvantage as some end users get confused by this, and answers indicate that users would prefer a standard desktop application.

The approach to programming in DHIS 2 is somewhat "heavy" compared to what can be done using other technologies. Users with experience from DHIS 1.4 point to how slow DHIS 2 is updated with new features on request compared to DHIS 1.4. Some participants with experience from scripting type of languages such as PHP and Python express a frustration about the coding principles of DHIS 2, and they generally find it to be a tedious process to add new functionality to DHIS 2. Combined with discussions failing to reach conclusions and the focus on the technology itself, the development process has manifested itself as a point of frustration for many users.

The database i18n is well received by users who intend to use this feature. Users with no interest in this feature see it as "just one more thing" they have to deal with, and the low score can by closer examination be attributed to this. There has also been several database issues related to the M7 release, where database i18n is included, that is unrelated to i18n, fostering some frustration among the users.

6. Discussion

My research objectives will in this section be discussed in light of the findings of my empirical study.

6.1 Assessment

I will start the discussion by doing an assessment of my field study. First, it's necessary to establish how to assess this process. My empirical base is somewhat unusual, as it's based on an application that's designed and developed primarily in a western country especially for developing countries often regarded as less substantial markets by commercial actors.

"Poorer countries would also benefit from technology which helped developers to produce systems for less substantial markets. The benefits to organizations are obvious: an opportunity to boost international market share, and simultaneously reduce risk by diversifying their served market." (Mahemoff and Johnston, 1998)

I would argue that corporate view of Mahemoff and Johnston (1998) still applies to some degree, as gaining international market share is of value for the HISP project. However, for HISP project, value is measured in terms of number of locations implemented, how the system is used locally in terms of local empowerment and its sustainability (ref. 4.1). Hence, to measure the success of the Vietnamese localization process all of these aspects should be considered. The literature indicates that there has been a lack of focus on the use of technology in commercial projects during technology transfer processes:

"The objectives and interests of technology source and recipient are often mismatched. In particular, the Western source of technology may seek an immediate profit from the sale, but have little interest in helping the developing country recipient make the technology work." (Baark and Heeks, 1998, p. 3)

I have through action research been colored by my own experiences, and in trying to assess this I have relied heavily on the survey responses and other interviews. At the time of writing it's one year since I concluded my direct participation in the implementation process, hence the Vietnamese development node surveyed can provide some indication of the sustainability and the long time effects of the localization process.

6.1.1 Implementation

There were issues with both the localization and internationalization effort. Starting with the positives, the final solution provided for HCMC was mostly well received. An interview with a Vietnamese developer acting in a support role in HCMC revealed that the first district we implemented DHIS 2 in (Tan Binh) now have developed their own datasets and reports for "in house projects". This is in accordance with HISP's goal of local empowerment and the hierarchy of standards (ref. 4.1). From the perspective of the end user we were able to meet all the requirements, though some of the users' patience has been tested due to performance issues with the software (ref. 5.5.1). We did a few tweaks to the services to improve performance, but the amount of memory required by the DHIS 2 application itself was difficult to limit. It was very important to us to show that we had a more sustainable solution with a support team this time around, hence the first pilot district was supported a total of five times after implementation.

6.1.2 Localization

Even though the implementation in HCMC seems to be a success, there is a lot to learn from the process leading up to it. The installation procedure had to meet a strict set of requirements never faced by DHIS 2, some of which probably were avoidable, but given DHIS history in the country there was a tension here that we didn't want to challenge. Due to time constraints we had to cut a few corners during the localization process. This resulted in an installation package that was highly particularized for the Vietnamese context. This installation package was complex to setup, which hindered its diffusion to other development nodes. A collection of 8 Windows batch scripts and executables had to be executed in a correct sequence by the installer. After a compilation process through the Installer2Go software this results in one "closed" executable file, hence a recompilation is necessary in order to modify the installation.

It should be mentioned that the HCMC approach was used in other countries, as the Vietnamese team was made responsible for packaging installation solutions for African countries. This bears a lot of resemblance to the centralized way the DHIS 1.x is packaged (ref. 4.2.2), and it's far from the distributed development goal of the DHIS 2 development (Nordal, 2006). The closed nature of the installation package also meant that it had to go through the packaging process again for every little modification. Slow internet connection and different time zones made this a very tedious approach, which was abandoned quickly. Such close interdependency between sites is also strongly discouraged by Herbsleb and Mockus (2003), who emphasizes the need to decouple development nodes from each other as a strategy for effective GSW.

In the DHIS 2 case, local time pressure and strict requirements introduced locally lead to a particularized installation procedure based on a collection of Windows batch scripts that were intricate set up using the installer software, which ended up being locked into the Vietnamese context. Interestingly, actors in charge of the Indian localization of DHIS 2 point to this as one of its major flaws in my survey, which could have been resolved by paying more attention to generalization during the development of the installation procedure. A more general solution would not only have made it easier to create installation packages locally in Vietnam, but also facilitated a global diffusion of the technology. There were also efforts to create an installation package from the student courses at UiO, but nothing usable materialized from this.

Prior to my field trip I was one of the people who advocated a lot for more sharing at the development nodes, but after being a part of it I have now gained an understanding as to why this often doesn't happen. My empirical study suggests that creating general solutions that are applicable on a global scale will require a considerable larger effort than creating specific solutions. I was surprised as to the degree of this. The need for refactoring of the resource editor program (ref. 5.1.2) can also to some degree be attributed to this effect. It wasn't needed in a functional sense, as all the necessary functions existed in the original version, but it is interesting to see how the code quality

suffered locally under time pressure. The refactoring was primarily something I wanted to do to provide sustainability of this application to allow other developers to continue the development of it when I leave the project. The key to avoid such problems may be as simple as just better planning locally, but this case illustrates how easy it is to loose the global perspective when working under time pressure locally.

6.1.3 Standards

During my empirical study it became clear that the lack of standardization of locale definitions (ref. 2.9.4) can pose challenges. Sun Microsystems (2006) recommend the locale definition to be used in Java to identify encoding, platform, language variants or dialects, such as the variations “B” (Bokmål) and “NY” (Nynorsk) of the Norwegian language, which both translate properly using the Java API (Figure 11). The use of variants in the previous example is both superfluous and incorrect. Superfluous because a standard incorporating these languages already exist in ISO 639-2 (ISO, 2007d), respectively by the codes “NOB” and “NNO”. Secondly, it’s incorrect because the languages are classified as two distinct languages and not variations or dialects of one single language (Vikør, 2001). The emergence of Unicode as the industry standard of encodings (ref. 2.11.6) further suggests that this field is somewhat superfluous, as explicit declarations of encodings should not be needed. Hence, I would like to emphasize the need for projects individually to standardize the usage of the variant property. Additionally, if this field is needed to fully define the language properties, as it proved to be to make a distinction between the two official Norwegian languages using Java, it should be used with caution as not all locale specifications will have the space in which to include any additional definitions.

As I have demonstrated in the example of above, the lack of standards in the locale definition require project individually to define the usage of locales. XenCraft (2005) points to how this is hurting the software industry in terms of standardization and interoperability between software projects. In the case of DHIS 2, we had to deal with a large number of computers in developing countries and chose not to “trust” the locale setup of the operating system of each computer individually. This led us to manually

threat the locale parsing in a two level hierarchy of locales in two individual dimensions, one for the user-interface and one for the persistent data. These choices are described in my empirical study, and visualized in the following figure:

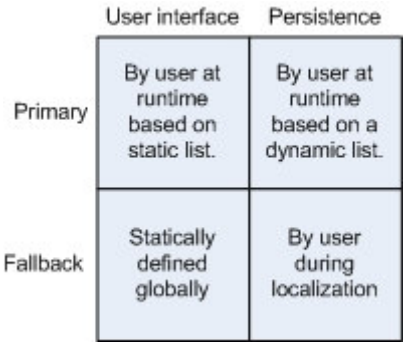

Figure 15: DHIS 2 locale matrix

When developers add new text to the user interface, they add them as strings in the language of the fallback locale. Hence, these strings will always be present and will always be what the system ultimately has to fall back on if no other translations are found. Similarly, this is the case for the translations of persistent objects, as users add an object to the database it will be bound to a particular locale. The solution for object persistence is as described in section 5.3 more complex in nature, but both solutions can benefit from an extended locale hierarchy. While a two level hierarchy is a workable approach, it is also very limited. For instance, an African country that uses a variation of French would most likely prefer to fall back on a general French translation before the defaulting to the English translations. This limitation did however not cause any direct problems in the case of DHIS2, but it is something to be aware of.

6.2 Software design and modularization

The architecture of applications is decisive in its ability to localize to specific contexts, and architectures based on a modular design are widely accepted as an enabling factor to support such local variations (Nielsen and Nhampossa, 2005). Internationalization is commonly regarded as being a very complex process to apply to an existing system and something that should be a part of the initial design (Sun, 2007a). Internationalization was not a part of the initial design of DHIS 2, but its modular design contributed to ease

the implementation of internationalization capabilities. The flexibility of the frameworks (ref. 4.3.2) allowed a separate module to be developed and aligned with the DHIS 2 application, with very little impact on the architecture of the system. A negative effect of modularization is that the amount of code will necessarily increase, because of the code required by the components to interact. There is however very little empirical evidence that there is a significant performance tradeoff in terms of runtime speed of the application in the enabling of modularization (MacCormack et al., 2006). Additionally, modularization can be used as a means to cope with the complexity of software design (ibid.). Several of the localization tasks, such as adapting the application to a different screen resolution and various GUI tweaks supporting requirements particular to Vietnam (ref. 5.2.5), were straightforward to perform during my empirical study because of the modularized nature of DHIS 2 (ref. 5.2.5).

An important aspect of modular design and software design in regard to localization is how external services are supported. The persistence in DHIS 2 can by the use of the Hibernate framework support a vast amount of database systems. This type of flexibility can in many cases lower the threshold to get into new countries, as the government in some cases may dictate the database servers (Nordal, 2006). This abstraction will however have a cost in terms of performance, as additional tasks will be performed in the translation done by Hibernate. Such flexibility is also present for the other main service required by DHIS 2 (ref. 4.3.4), as most Servlet containers can interact with the most commonly used web servers. The reason for such requirements can be political/ideological, because of former investments or even security related.

"Some projects have a growth rate never seen before in similar-sized closed software projects. That can be explained partly through the possibilities for participation offered by the Bazaar model and partly because of an efficient technical and organizational modularization." (Robles, 2004 p. 13)

Research suggests that the enabling of flexibility is one key advantage of OSS. The availability of services, and the lack of licensing issues and bureaucracy in OSS (Weber,

2003), gives OSS an advantage over more strict licensing in regard to flexibility in localization processes. As presented in a survey conducted across the Asia-Pacific in Wikibooks.org (2007), "freedom to develop" was the primary reason for choosing to localize OSS instead of using proprietary software and modularization is a contributing factor in the enabling of such freedom. DHIS 2 facilitates freedom to develop by not only allowing what modules to be included to be customized at the local level, but also by allowing local teams to develop and introduce modules to the system during localization.

"The open source process has the potential to empower developing country end users to customize applications for the very particular needs that often arise in different settings, and allows, through use, the natural evolution of information technologies and systems within unique and specific contexts." (Weber, 2003. p. 21)

In the case of DHIS 2 it's still at an early stage of its development and future requirements that will emerge from governments in which HISP is involved are uncertain, which is a viable argument to keep a flexible design and avoid locking into a less flexible technologies. Less complex systems can benefit from using more accessible technologies such as script languages, but the process of enabling localization across contexts globally is arguably in itself complex. Hence, systems that need or may need localization features should plan according to this, and such planning did prove to be useful in the case of DHIS 2. The bottom line here is to establish the need for modularized architecture, both to support distributed development and to support local variations of a system.

Robles (2004) attributes the super linear growth of Linux, which evolved from a "hobby project" of a Finish programmer to a major operating system, to its strong modularization. I have observed a significant opposition towards the software design of DHIS 2 by some actors which is also reflected in the survey. Removing such layers would cause the amount of code to decrease, and hence also possibly lower the threshold for new developers to join the project. I argue that this would not benefit the system in the long run, especially in regard to localization capabilities. As I have presented, I did

require the flexibility of the frameworks and the multiple layering in the implementation of persistence-layer internationalization. Additionally, based on a study of the refactoring of the Mozilla project , MacCormack et al (2006) point to how modularization can increase the amount of contributors to an OSS project. This observation is however based on a "conventional" OSS project, where users join the project by their own initiative and are familiar with the communication tools and programming practices. My empirical base is in several respects different from that, as it was to a large degree based on staff that lacked experience both in regard to technology and the use of communication tools. Striking a balance between the complexity of the frameworks and the inclusion of less experienced programmers has proven through my survey to be a challenge. My findings go a long way in suggesting that the technology used in the DHIS2 project has been a major contributing factor in the project's failure to fully include a significant number of programmers from the developing nations.

6.3 Global development

The two primarily sources of requirements for DHIS 2 are the existing functionality of DHIS 1.4 and requests from development nodes in developing countries. Pollock et al. (2007) describe how the shift from a particular to generic technology corresponds to a larger extended community. The paradigm shift from the first generation of DHIS 1.x to the distributed development of DHIS 2 presents such a change, in which a larger development community has emerged. This has in turn led to a shift in the technical discussions towards a more public forum, where users and developers interact frequently. This presents new challenges to the development process, as the strive towards creating standardized generic solutions that serve the whole community and the need to serve the users individual needs often seem to be at odds. The participatory development philosophy in which HISP was founded on is arguably more challenging to achieve as the development network grows globally.

MacCormack et al (2006) point to how organizations with modular structures often develop software with a large degree of modularization. The organizational structure of HISP is highly modular (ref. 4.1), with development nodes on three continents.

Additionally, the facilitation of GSW was one of the key reasons for the development of DHIS 2, and I have in the previous section and in section 4.3 established that DHIS 2 has a highly modular architecture. MacCormack et al (2006) further suggest that an architecture with a large degree of modularization will draw more contributors to a project, but this has as mentioned largely failed in the case of DHIS 2 in developing countries. My survey points, as mentioned, to the complexity of the frameworks as a contributing factor in this, but this was not present in my field study as the Vietnamese development node proved to be able to cope with this complexity (ref. 5.4.2). Still, they are along with the other development nodes not able to provide any concrete development contributions to the project globally. In light of this, I will in the following sections further discuss distributed OSS development in the context of developing countries.

6.3.1 Communication tools

That the actors are able to communicate by the use of the online communication tools are arguably a prerequisite for GSW (ref. 2.5). I have presented several occasions where the western oriented open source project and a strict hierarchical structure in developing countries seemed to be at odds. This section elaborates challenges and solutions related to this in the use of mailing lists in my empirical study and the process leading up to a standardized issue tracker.

Mailing list

Edwards (2001) points to how contributors to OSS projects share a common frame of reference and how users by participating in the project gradually gain an understanding of this reference frame to be come a part of the community. The mailing list is the primary communication channel of my empirical base in terms of communication between developers (ref. 4.3.3), and the use of this communication forum is a prerequisite for integration into the development community.

"It is not possible, given the limited means of communications, to socialise people into the community and create that shared frame of reference, which that has a significantly different mindset". (Edwards, 2001, p. 18)

This assertion is to some degree reflected in my research. The integration of development nodes in developing countries with different mindsets proved to be challenging by conventional OSS communication methods. My research further indicates that this problem in relation to communication practices is two-fold. Firstly, the language barrier was, as reflected in the interviews conducted (ref. 5.4.2), a factor that contributed to raising the threshold for integration into the communication framework. This is however an overt and observable factor that can be measured, my research further suggests that a less apparent barrier against taking the initiative in public conversation was equally challenging. The basis for the reluctance to participate in the online discussions can, based on the interviews conducted, be translated to be caused by an unfamiliarity frame of reference used in the discussions:

```
(Vietnamese developer 1) ah, about the reason for not using the dev list at the beginning
we was not familiar with doing that, and still shy to do that and we didn't really
understand the system
```

Chat transcript 1 (ref. 5.4.2)

My research further suggests that relationships built during participatory research methods, such as action research, can be a used to overcome this threshold and manage socialization. The failure of establishing such communication practices at a partner university where less time was spent, and interviews conducted (ref. 5.4.2) further substantiates this claim:

```
<Question of reasons for the failure of establishment of communications with the partner
university>

(Vietnamese developer 1) first is the language problem, second is they didn't work with
us closely, so they were afraid of it, and they don't know who will answer, and how are
they, so ... (Vietnamese developer)
```

Chat transcript 1 (ref. 5.4.2)

In conclusion, while socializing actors with different mindsets into an open source community proved to be challenging, my research suggests this can be achieved through extended means of communication through participatory efforts.

Issue tracker

Initially, the JIRA issue tracker was used in the DHIS 2 project to manage bugs and handle planning of features for DHIS 2. The use of this application did however halt (ref. 5.4.4), as the developers in developing countries were reluctant to use it to submit issues (Nordal, 2006). Responses to my survey can be translated into the acknowledgement that an issue tracker is needed for the DHIS 2 project, as users generally felt that issues filed on the developers' mailing list died out without reaching a conclusion. The introduction of an issue tracker eliminates this problem, as the issues filed are listed until they are resolved by a developer. Hence, I conclude that both the developers and the users locally wanted such a solution. The reaction from the global development community when the new issue tracker was introduced further substantiates this conclusion (ref. 5.4.4). The development nodes almost immediately started submitting issues, which were automatically sent to the developers' mailing list by the issue tracker and used as a base for further discussion.

"The design of global information systems needs a persistent focus on the work practices at local level in order to achieve the intended support for the work, and in order to ensure that the system at least does not contradict the employees' own work practices. In addition, paying attention to the minor but locally relevant features of the system can go a long way in assisting the diffusion of such collaborative technologies within a global organization." (Joshi et al., 2006, p. 29-30) .

The initial solution of JIRA was based mainly on the technical interests of the core developers, as it was chosen during the initial stages of DHIS2 development before any other development nodes than Oslo existed (Nordal, 2006). The reluctance of the nodes in developing countries to use this tool resulted in the workaround that the nodes in developing countries filed issues directly on the developers' mailing list, which in turn

often got lost or not properly resolved. As indicated by my survey, this lead to tensions between the users locally and the developers globally. That such tensions emerged can arguably be attributed to the fact that the local working conditions were not taken sufficiently into account when an issue tracker solution was decided, and it does to some degree confirm the assertion of Joshi et al. (2006) that it is necessary to acknowledge and balance local working conditions when implementing tools to facilitates global communication.

I have little concrete empirical evidence as to why JIRA failed to adapt to local contexts while Trac seemed to align with the local working conditions, but my survey and empirical study suggest that this may have been grounded in the differences in technical complexity between the two tools. The level of technical complexities was pointed out by several actors as a major issue (ref. 5.5.1), and the motivation among all users to use such functionality have been established. There was a cost involved in this change of technology, as a more sophisticated issue tracker was replaced by one with less features in terms of issue tracking. This case suggests that it was necessary to strike a pragmatic balance between the technical features wanted by the core developers and a tool that all the users were able to use in order to achieve a workable standardized solution across the development network (Rolland and Monteiro, 2002).

6.3.2 Digital divide

Performance was highlighted by the respondents of my survey as one key issue, which suggests that there already is a significant digital divide present in the project. In the case of HCMC, the technical level of the computer equipment proved to be significantly lower than what we would expect in the western world. They were for instance still selling new computers with CRT monitors as opposed to LCD, and a quick survey of the most used web shops in HCMC proves that this still is the case as of June 2007. The standard of a 1024x768 as the lowest supported screen resolution seemed like a safe decision when it was made in a “closed” environment in Oslo, but supporting resolutions significantly lower than this was necessary in the localization process for DHIS 2 in HCMC (ref. 5.2.5).

The technical challenges of introduced through the digital divide were, as mentioned, for the most part manageable through the flexible structure of DHIS 2. However, by adopting the perspective of Gurstein (2003) on digital divide, I can point to several cases where there was a lack of effective use of available technology. I have in the previous section emphasized a division in the use of the communication tools, which had western cultural values inscribed, proved to be challenging to introduce into the Vietnamese context. This posed issues that were difficult, but possible to overcome through participatory learning processes. Another key example of such cultural conflicts is the denial of the use of a centralized online version of DHIS2 in the health districts in HCMC (ref. 4.4.7), where health managers refused to provide access to the Internet to the other health workers. The technical infrastructures were available and the solution was established, but the social structures did not allow the effective use of this solution. Hence, we ended up using local installations that were both more complex in terms of maintenance and in terms of importing and exporting data in the hierarchy of the health sector (ref. 5.2.7).

Another case of failure to integrate a local project globally emerged from India, and it resulted in a highly particularized solution that was created to support a local requirement (ref. 5.4.3). It can be argued that this is a type for "forking" (Kogut and Metiu, 2001), as it shares a common data repository with DHIS 2 and it emerged from within the same community. This project is based on the same programming language as DHIS 2, but failed to align with its frameworks and development practices. This did, in turn, based on the introduction of changes to the structure of the database, lead to a tension between the developers globally and the local team. A core DHIS 2 developer expressed concern about how this particular situation could hinders the development, and how it potentially could grind the development to a halt (ref. 5.4.3). The common reliance on a database system in this case illustrates a consequence of failure to integrate a local development node in the global development process, both technically and socially.

As pointed out in my empirical study, and observed by both Øverland (2006) at a university and Nordal (2006) in a software developing company (ref 5.4.2), there was a

lack of initiative among the Vietnamese developers when a task was completed. The Vietnamese people have a history of suppression (ref. 4.4.3), and this is in my interpretation reflected in the social structures and in work settings where people generally seem to expect to be given tasks by a higher authority. This is at odds with the OSS communities, which to a large degree rely on the initiative of actors to actively seek programming challenges to integrate with the communities (ref. 2.10.3). While a digital divide in terms of availability of technology existed in my empirical study, the divide between cultural factors of a developing country and an OSS community presented considerable more challenges in terms of effective use of the technology.

6.3.3 Generification and global diffusion

The results of the survey lead me to conclude that the DHIS 2 users in developing nations generally don't feel that their requests for new features are being met often enough (ref. 5.5.1), but we have been able to work towards generic solutions on several occasions. The enabling of language translations of the user interface (ref. 5.1) is a simple example that illustrates this point, as the alternative would be to have individual branches for each local implementation with their own web modules.

Another case where general solutions were reached successfully is found in the implementation of export routines in DHIS 2, which emerged from a requirement from India for the system for the ability to generate comma separated text files (CSV) with aggregated data and calculated indicators. Functionality was developed that performed this task, but it used a fixed format and was not configurable to any extent, and therefore was only usable for India (ref. 5.4.1). Pollock et al. (2007) emphasize how participants became more accommodating towards collective requirements by spending time on getting to know the size and complexity of the task at hand, and this proved to be the case for DHIS 2. After the South-African workshop where all major development nodes and health staff were represented, a more generic solution was designed and developed. It is based on a template architecture where an unlimited amount of templates can be developed as plugins, which all appeared in a drop-down list in DHIS 2. The development of such templates can indeed be regarded as localization for specific

contexts, but it can also be used to write templates to support export of international standards that can benefit the project globally.

This process of generification emphasizes the importance of aligning interests within a community (Pollock et al., 2007); by describing what the feature needed they aligned the solution with demands from other users and the global community was able to benefit from the initial requirement. Additionally, the quality of the general solution negotiated through personal meetings of the workshop is in contrast to the particularized solution that emerged from GSW discussions. While this points to the challenges introduced through the limited communication channels of distributed development (Herbsleb and Mockus, 2003), it also substantiates my conclusion that a digital divide exist in the effective use of communication technology in the DHIS 2 case.

6.3.4 Effects of localization on global development

Decisions made during localization processes can also to a large degree influence the direction of the global development process. I noticed tendencies towards database specific solutions being introduced to DHIS 2 when we started the preparations for implementations in HCMC. It is possible to execute native SQL statements through Hibernate, and hence be dependent on a specific relational database. At the time, MySQL was the only DBMS for the pilot districts in India. With a hidden agenda of keeping the database abstraction I ended up pushing for PostgreSQL, which is widely regarded as a robust and stable DBMS. It's also a DBMS that conforms to international standards and we immediately discovered several bugs in our code after it was introduced, such as the use of reserved SQL keywords (ref. 5.4.2). There was a bit of a struggle to get this through locally in Vietnam, with opposition from the former OutSoft employee who only had personal experience with MySQL. The Vietnamese government had no preference on this matter, and we kept this struggle within our team. With support from the rest of the development team I was able to get this through, and PostgreSQL is now the standard DBMS of DHIS 2 in Vietnam. This is an example on how localization can shape the global development process through the use of abstractions in software design, and how flexible standards (Braa et al., 2007) are empowering to local users. As long as the

database server implements a common SQL standard through a Hibernate dialect, users are able to select any database system and take advantage of features of that particular database system locally. In contrast, strict standardization and control could have been exercised globally by forcing the use of one database server.

The description of Raymond (2000) on how multiple views of a system can tame complexity provides a useful perspective on the database abstraction (ref. 2.10.4). The localization process in Vietnam caused a different DBMS to be introduced into global context, which resulted in the discovery of several SQL standard related bugs (ref. 5.4.2). Additionally, the diversity in the tools used among the developers in the DHIS2 case has several times uncovered new bugs related to specific platforms. DHIS2 is currently being developed from Windows, various Linux distributions and Mac OS X.

6.3.5 OSS adoption developing countries

I will now discuss the motivational factors of adoption of OSS in developing countries as presented by Weber (2003) (ref. 2.10.5) in light of the findings from the DHIS 2 case.

Independence: While the use of open formats in OSS in it self provides some degree of independence, the independence of OSS is compromised if developing countries are unable to participate in the development. My study indicates a high degree of dependence on the global development community to provide functionality, and that actors in developing nations assume roles closer to users than developers. This arguably limits the potential for the development of local competence that can serve the local economy. The DHIS 2 case does however show that significant independence is attainable in regard to local maintenance, as nodes in developing countries are able to exploit the flexibility provided by OSS in localization processes.

Security and autonomy: Security is by many regarded as a key argument for choosing OSS over commercial software, because of the rate in which bugs are discovered and fixed (Raymond, 2000). This is however hardly an established fact and a subject of considerable debate (Chelf, 2006), which is not within the scope of this thesis. The

question is then, if the availability of the source code enables governments to provide additional security, and I have seen very little evidence of that happening. The modularity and the responsive communication in the DHIS 2 project did prove to be contributing factors in the enabling of the login security functionality of the online solution (ref. 5.2.7), but again this was developed globally and reflects the strong dependency on the global community. Additionally, I have no evidence to support that this would have been supported any slower in a commercial projects, but a high degree modularity is widely recognized as an argument for the adoption of OSS (Robles, 2004). While the availability of the source code in principle enables independence and autonomy, it is compromised if the nation is unable to exercise the right to modify the code and engage in the development community.

Intellectual Property Rights and Productivity: The case study from a developing country that has announced a devotion to use OSS revealed no such commitments in practice (ref. 4.4.7), with the exception of the DHIS 2 application. As widespread use and acceptance of piracy of software was observed, the price argument for OSS was not applicable. No ideological base for OSS observed, and according to my observations both governmental officials and technical staff seemed indifferent and ignorant to the adoption of OSS. The flexibility provided by OSS did however assist in meeting requirements introduced locally during localization, and some degree of innovation emerged from the localization process (such as the date selection widget, ref. 5.2.6). However, as mentioned, the development nodes in the developing countries have showed very little capability to participate in the development of OSS without direct local participation from western developers. Hence, in the context of my empirical study, the amount of diffusion and innovation from such nodes were very limited.

6.4 Linguistic translations

The DHIS2 approach to translations has proven to be problematic, as the translators may need to know the context in which the sentence is used. The resource editor (ref. 5.1.2) does not provide any contextual information except the existing translations references. My research strongly suggests that performing translations implies both technical

knowledge and adequate knowledge of the domain in which the software translated is to be used. Experience from the DHIS 2 case indicates that handing over such software along with translated content to an independent translator is by no means sufficient. The translators also need to be aware of the software in general and what context it is to be used. In Vietnam we had to change specific translations translated by the Vietnamese IT team up to 3 times to make it intuitive to the health staff (ref. 5.2.6). Complexities such as ambiguities in languages can cause loss of precision during translations, an example of this was mentioned during the introduction of open source (ref. 2.10) and the word *free* in English contra Roman languages (Fogel, 2005). Another example on how this caused a problem in the implementation of DHIS is highlighted by Nhampossa (2004) in the context of a translation to Portuguese in Mozambique:

"Language problems were the most critical due to lack of understanding of the terms visualized on the user interface and linked to specific functions of the DHIS. For example, the string data element was translated as elemento de dados, but on testing we found that the meaning was distorted by the pure text translation performed by people lacking expertise in technical health terminology. Variável for variable is in this case the correct translation according to health workers." (Nhampossa, 2004, p. 10)

This highlights the need for verification of linguistic data. Further, it suggests that a participatory action research approach of having pilot phases initially during implementation efforts, and having verifications of such translations as an important part of those phases as an ideal framework to include such testing. There is also, as Nhampossa (2004) points out, a need to strike a pragmatic balance between the software design and the task of performing translations. The tightly coupled architecture of DHIS 1.x allowed translations to be performed directly in the application at the local health sites, while as demonstrated in figure 12 this process is much more tedious in the case of DHIS 2. This is in contrast to the object translations described in section 5.3 that are performed directly in the application, but it also very different in nature as the translations emerge from a central repository and is tied to the application and not its data. I conclude based on my survey (ref. 5.5.1) that that the users are generally happy

with this solution, but I would argue that the separation of language translations into external language packs that would relieve the system of recompilation of the main artifact to update translations is a more pragmatic approach. However, my main point here is to emphasize the requirement of linguistic testing of a system along with personnel fully proficient in both the language and the technical terms of the localized application.

The process of language translation of software will arguably always to some degree be a chicken and egg problem (Wikipedia.org, 2007), should the software used to translate the software be multilanguage enabled as well? Whether it's a good decision to enable these types of translations into the target language is likely to depend on each projects individual need. It may make sense in many projects to translate it into a few of the worldwide languages such as English, Spanish and French. In the case of DHIS 2 we decided that this was not needed, mainly because all development nodes have English speaking personnel. As a compromise all of the most essential features were labeled with both icons and text, and the interface of the resource editor was kept very simplistic (ref. 5.1.2).

An interesting approach to translations currently being tested for DHIS 2 is to use automated machine translations as a first phase of the translation process. This may be a dangerous approach, especially in regard to subjects discussed above. However, with proper testing I suspect that such translations can be used as a poly-generical approach (Pollock et al., 2007) to translations, creating neutral templates that can be localized for specific locations implementation that language. It will however still be a requirement for translators to be proficient in at least one of the verified reference languages and to get the technical terms verified.

6.5 Illustrations and user-interface

Recently the mergers between the global Norwegian oil companies Statoil and Hydro launched a collaborative project which has the objective of designing a common brand for the two companies (Hydro, 2007). The process of finding a logo that would work

across all cultures was identified by analysts as one of the most challenging parts of this project, which has a budget of about 30 million USD. This emphasizes how seriously cultural aspects are taken in the corporate world, and research from IBM (ref. 2.11.5) indicates that this to a large degree also applies to software development. Graphical illustrations do not communicate the same message across cultures, and they may unintentionally be offensive to a target culture. One example on of how illustrations fail to communicate the same message across contexts is how different cultures associate the meaning of colors. Red may for instance be an intuitive color to use on a symbol for an error or warning in most western countries, but in China and other parts of Asia the red color is associated with happiness and good fortune (Dix et al., 2004).

My impression from my own observations in Vietnam is that this issue may be a bit exaggerated. Users generally don't seem to expect such culture dependent aspects implemented in computer systems, as they are already used to run systems with a western oriented angle. This does however not mean that there isn't much to gain by implementing such functionality, but this should be done with great caution. For instance changing informative confirmation messages to use a red icon may be seen as something negative because of how this has been formerly inscribed by other systems, even if this matches a traditional cultural preference. This view is supported by Mahenmoff (1998) who emphasizes the need for real user-testing, before such functionality is implemented. Cultural preferences can however shorten the process of establishing such solutions by providing initial direction, and as pointed out by Reinecke and Bernstein (2006) software localized in regard to cultural aspects can increase efficiency dramatically. Their research further suggests that cultural aspects have greater value when users have no experience with computers (ibid.), which substantiates the viability of the pragmatic and general approach of the OLPC to internationalization of graphical illustrations (ref. 2.10.8).

One way of handling visual illustrations is to "go back to basics" and appeal to basic human emotions. Such as the OLPC projects approach of using a picture of an eye instead of a camera for the computers built in camera (ref. 2.10.8). Reinecke and Bernstein (2006) describes based on a case study that tailoring culture adaptive software

as prohibitively expensive, and points to technology such as artificial intelligence as an alternative to manual localization. This may represent the two extremes in localization processes, from the most general and static to technology that is dynamically able to adapt to a specific culture. I will continue this discussion by comparing pragmatic methods of applying internationalization to illustrations containing text.

Abstractions can be applied to graphics by the use of layering, separating the text from the graphics. With focus on web-based projects, I have outlined two different methods of achieving such abstractions (ref. 2.11.5):

- Runtime static by the use of imaging software and layering.
- Dynamic by the use of text positioning over the image by the use of CSS technology.

This would make the text directly available to search engines and relieve the localization effort of image editing, but some browser has serious issues with this technology. Especially older versions of Internet Explorer handles CSS positioning badly by not conforming to the official CSS standards (Gallant and Bergevin, 2007). Since the big majority of internet users still use Internet Explorer (W3, 2007a), the approach using CSS is arguably too much of a risk to take for most projects. Considering the installed base of Internet Explorer the static image alternative is emerging as the only viable choice.

It can hence be observed based on this short discussion that the technology to enable adaptive solutions proved to be insufficient, and that static abstractions and general solutions emerge as the viable alternatives for a pragmatic internationalization solution also in this case (ref. 5.1, 5.3). Though cultural effects such as the perception of colors should be considered, I argue that they should be considered in a bigger picture of the effects of culture specific images across contexts. This perspective on these issues are substantiated by IBM (2007), who suggests that the cultural factors may have a negative effect on the willingness to learn to use an application or web-site of perhaps as much as 15% of the users, and emphasizes the need to establish mechanisms to verify that graphical illustrations work across cultures (Mayeur, 2007).

6.6 Encoding and input systems

Encoding received very little attention during my field study, and the primary reason for this is that we had very few issues with it. Unicode (ref. 2.11.6) is used consistently throughout DHIS 2, where characters are encoded using UTF and written as UTF escapes ("\uXXXX") to external persistence services. The transparency added by using technology such as Java that fully supports Unicode was an advantage in our case. Unicode assisted in handling details such as multi-line support in properties files by writing such line changes as Unicode escapes, which can be an issue in single line organized properties files over multiple platforms (key=value on each line).

We don't have to look further than the first implementation of DHIS 1.3 in Vietnam to find a project that had several issues with lack of support for encoding of characters. DHIS 1.3 is programmed in Microsoft VBA 6, which implements the ASCII CES, and needs to be extended to fully support Vietnamese characters (Øverland, 2006). Unicode is the only encoding scheme that attempts to adapt all characters encoding in a uniform manner (UnicodeConsortium, 2007). Projects that implement or plan to implement any internationalization capabilities should make a conscious choice about encodings. Unicode is evolving to be the de facto standard of encodings, and at this point it seems difficult to justify choosing technology that doesn't support it.

When we started translating in Vietnam a different type of encoding challenge came to our attention. Even users in the technical team had serious misconceptions about the relation between fonts and encoding. This became very apparent when they did translations and translated using various legacy "font translations" systems, meaning that only how the characters are rendered at screen are changed by changing the font. Thus, the underlying codes are left unchanged and the information later displayed in the web application will be incorrect.

6.6.1 HTML and Unicode

A completely different issue that can potentially be serious in regard to encoding is the use of HTML codes ("&#2744") instead of using a proper CES such as UTF stored as

escapes (“\uXXXX”). This is especially “dangerous” as it will appear correct to the user using a web browser, but the system itself will be unable to encode the characters properly. Hence, if the system is required to process the text, for instance for sorting or to check name constraints, it is likely to not produce correct results. Additionally, conversions between these two formats are not trivial. Getting non-technical users under a great deal of time-constraints to realize such issues has proven in the case of my empirical base to be quite difficult, and currently DHIS 2 has several resources persisted using HTML codes instead of UTF escapes. This reduces the modularity and transparency of the software design, as the HTML technology only should be situated at the presentation layer (ref. 4.3.1). This does however not currently pose any serious threats in the case of DHIS 2, but I would like to stress the importance of being aware of such issues.

6.6.2 Input systems

The actual input of characters are often handled through software such as Baraha (used in India) (Baraha, 2007), which need to be configured correctly to output the right encoding. An alternative is to integrate a complete input system into the translation program. This is however a comprehensive task and the users are likely to have to learn new input methods, which my observations indicate that they are reluctant to do. Some of the script inputs are quite complex and they often require as much as 4 keystrokes for a single two-byte character. We have had several issues with these input systems. An Amharic (spoken in Ethiopia) specific input system caused Firefox, which is the browser DHIS 2 is commonly bundled with, to completely stop working. The reason for this was a bug in the input software itself, which has a closed source code. OSS may hence have to undergo compromises in order to support these types of software. In this specific case the issue was solved by using a different browser, but this uncovers a serious challenge for internationalized software development in general. As the users often are already locked into input system software, the new software introduced will have to be flexible enough to align with this software. After working with encoding issues in DHIS 2, my impression is that there is a serious lack of adoptions of standards in this area. There are however exceptions to this. The OLCP initiative is a project that is in charge of both the hardware

and the software. It is designed for users that have little experience in the use a computers, which provides an opportunity to standardize the input system on a global scale using OSS technology (ref. 2.10.8).

6.7 Internationalization of the persistence layer

In this section I will discuss the internationalization concept for the persistence layer in software projects that was presented in my empirical study (ref. 5.3). I have demonstrated that it is possible to implement a generic translation framework for objects and align them with an existing system as an aspect without any modifications to the objects being translated. The use of the term internationalization may be misleading here. This concept does not include any other functionality beyond translations, and hence the term object translations probably would have sufficed. This can however be confused with other types of object manipulation, such as casting, so I chose to include internationalization when denoting the concept. I suspect that this is especially intuitive to programmers who often regard internationalization as synonymous with language translations (ref. 2.7.3). It's tempting to borrow a term from the ORM methodology (ref. 4.3.3) to describe this concept and use *transparent internationalization,* as the POJO structure of the objects is kept intact.

6.7.1 Technical considerations

This concept was developed based on requirements from DHIS 2 and the case from India (ref. 4.5). The case includes a requirement for the database to have data for one locale only, and a wish to minimize the impact the introduction of this concept would have on the system. These factors contributed to a flexible and generic solution that may have a cost in terms of performance. Examples of this include the field length needed for translations, which is the value property in object description 3. Field lengths that contain internationalized strings will always have to be at the size of the longest (in terms of character space) internationalized element, which introduces an overhead for all fields that are shorter than the longest element. Additionally, the loose coupling between translations and objects also causes field values to be stored twice, on the object itself and in the translation tables, further increasing the database size. Although my tests with

DHIS 2 didn't reveal any significant loss of performance due to any of the issues mentioned, other projects may have performance issues related to this.

I would like to point out three specific technical problems that occurred during the process of enabling translation on the persistence-layer. These problems are to some degree specific to the technology used in DHIS 2, but they may very well also apply to projects using different technology.

Lazy loading: Some ORM frameworks, such as Hibernate, uses a technique often referred to as lazy-loading (ref. 4.3.2), which implies that object relations are not loaded until they are requested. This may potentially pose a problem when working with objects in need of internationalization, since these objects are loaded by the framework itself and not through the services of the application. In DHIS 2 we solved this issue by registering event handlers at the persistence layer (ref. 5.3.6).

Flushing: Another potential problem with ORM frameworks is related to how sessions are treated. In some cases an ORM framework may cause all objects to be saved, without specific calls for this from the application itself. Such events are commonly referred to as "flushing" and cause modified data to get saved, which will include internationalized properties. This is a problem for DHIS 2 because of the requirement to keep the database with information for one locale consistently (ref. 5.3.2), and it can be handled by event handlers. For other projects without such requirements this may not be an issue at all, but it is important to be aware of. It can, as I has experienced, be a source of confusion during debugging processes and testing.

Circular dependencies: If a project using a three-tier architecture chooses to implement the i18n translation procedures on the persistence layer, circular dependencies between modules may occur. In DHIS 2 we had to implement the translation process at the persistence layer because of the lazy loading issue (mentioned above) and we handled this by registering an event handler through a method invocation done by Springs' dependency injection (ref. 5.3.6). This method invocation could be configured at the

service layer and hence register itself at the persistence layer. Some transparency is however removed using this approach, and it's clearly not optimal. This is just a hack around the issue of circular dependencies, which is an important principle applied by Maven (4.3.3) for safe build processes and good programming practices. Further, this issue points to a conflict between the cross cutting concerns of an aspect oriented approach and traditional hierarchical object oriented programming in relation to module organization.

6.7.2 Scope

I have, as mentioned, not done any further research into the scope of the concept presented. My demonstration from DHIS 2 shows that it works with somewhat unusual requirements in a global context, but this approach would arguably get problematic if the i18n enabled properties need service layer, or even more complex – core, processing. The DHIS 2 case suggests that this approach can benefit many projects, for instance by providing translations of data to assist in a global debugging process (ref. 5.3.1). The concept presented binds all translations to individual objects, and may hence not be very usable for comparison with centralized registers such as ICD codes (WHO, 2007a). That was however never the intention of this solution, and DHIS 2 has a dictionary based project dealing with this. The strength of this concept lies in how it can generically be aligned with an existing solution with an installed base.

6.7.3 Generification through aspect orientation

The flexible nature of the DHIS 2 platform allowed us to develop this concept as an aspect oriented solution. By keeping a global development perspective, we avoided design choices that could have contributed to a particularizing of DHIS 2. This particular case was dangerous because it could have locked the Indian installation into a different database design and service layer implementations than the rest of the users. We had already seen some movement towards this by the addition of an "alternative name" field to some of the objects in the database, which was only used in India and not recognized by users globally. The concept discussed in this section made the alternative name field obsolete, hence actually leading to less particularizing of DHIS 2.

It is now a feature that all users are able to use if they choose to do so, but as my survey revealed (ref. 5.5.1), users who don't need it are not very excited about it. The inevitable need to define a fallback locale and to run the automatic upgrade procedure was seen as "just one more thing they would have to do" on top of an installation procedure they already regard as too complex. By using the cost term as defined by Rolland and Monteiro (2002), this can be seen as a cost that users that do not intend to use this functionality will have to pay in order for the DHIS 2 application to keep a standardized persistent structure globally.

6.7.4 Theoretical considerations and summary

The introduction of this concept to DHIS 2 does to some degree confirm the theory of Kersten et al. (2002) that an aspect oriented approach is highly suitable for internationalization purposes. The amount of cultural aspects captured by this method is somewhat limited, but language translation is, by far the most used internationalization method together with date and time formatting (Esselink, 2000). Tate and Gehtland (2004) suggest that it will take a full blown AOP language for the language to get widely adopted, but points to how it is already coexisting with OOP through frameworks such as Spring. I have demonstrated that the most widely used internationalization functionality can in fact be adapted to an existing project using AOP principles. I suspect that separating and generalizing culture specific core functionality in accordance with the theories of Kersten et al. (2002) would be a viable alternative to achieve internationalization at this layer if more advanced business logic requires localization, but generification work (Pollock et al., 2007) should be considered before deciding to isolate culture specific functionality that can lead to particularized solutions.

6.8 Validity of research

The empirical base for this study is to a large degree based on my own experience from a limited amount of locations. The development of the main artifact is centered among Norwegians who have their base in Oslo, but I would argue that the development process still keeps a global perspective. The users do play a significant part in the online

discussions by requesting features and help, and all of the Norwegian developers and coordinators are very rarely situated in Oslo at the same time.

The organization of the DHIS 2 project may resemble the structure of a commercial project more than a traditional OSS project. The Vietnamese development team from my case study was hired by the project, which possibly should have allowed the central management to dictate tasks. My data do however suggest that this wasn't the case. As the team was hired for a relatively low amount of money, they felt that they had grounds for negotiation of working tasks. Persuasion was necessary in order to move from the implementation work, and a significant effort was needed in order to integrate this team into the global development network.

As indicated by Avison et al. (1999) it is necessary to define research criteria prior to conducting action research. There is a clear risk of action research resulting in nothing but consulting work, or even action without research or research without action (ibid). I would argue that neither of these two alternatives are the case in my research. Although I acted as a technical consultant towards many in the medical community in Vietnam, I was able to carry out the research and get the results needed. I do however agree that the enabling of translations of persistent data bares less resemblance to the participatory nature of action research. It was motivated by a requirement that emerged from India and all important decisions were discussed publicly, but the majority of the actual research and development was carried out in Oslo and for the most part by me personally.

Language barriers did however (somewhat ironically in the context of my study) pose challenges during this study. The staff at the health stations and hospitals where we performed implementations was not proficient in English, and I had to rely on translations from the Vietnamese team to make any sense of what was said. The close collaboration encouraged by AR did hence arguably suffer from this; as such relationships were challenging to build using actors from the Vietnamese team as interpreters or just by using body language. I do however think that we made the most of

this situation, and this claim is further substantiated by the success of the implementation (ref. 6.1).

The limited scope of a master thesis allowed me to highlight a number of aspects of the complexity surrounding the global development of internationalized solutions. Through action research methods, I was able to gain unique field work experience and insight into the localization of information systems in a developing country and the establishment of a local team to support such solutions in such contexts.

7. Conclusion

I have demonstrated how a Health Information System (DHIS 2) was internationalized and localized in the context of a developing country. Flexibility was one of the key reasons for the development of a second generation of DHIS, both in terms of deployment and enabling global development. In the context of an action research study, I have looked at both of these aspects. I have pointed to a shift of focus on internationalization and localization as means of just isolating culture specific data, to a broader perspective on how organizational structures and technology can be adapted to support such change (Joshi et al., 2006, Kersten et al., 2002, Pollock et al., 2007, Rolland and Monteiro, 2002).

Primary research objective: *Explore challenges in internationalization and localization of open source software in the context of developing countries, with an emphasis on:*

- *The development of internationalization solutions and the establishment of supporting infrastructures.*

It is well established that internationalization traditionally is a process in software design of extracting and abstracting overt cultural elements (Kersten et al., 2002). I have demonstrated how an internationalization solution was developed in the context of a HIS, and discussed how abstractions apply to visual illustrations as well as text and date formatting. My research suggests that a key challenge in the development of such solutions is the necessary alignment with existing input software, which often seems to be of low quality, but with a significant installed base that makes them difficult to replace even if better alternatives are available.

I have advocated the use of modularity and generalized solutions, and illustrated how flexible architectures are enabling in avoiding particularized solutions (Pollock et al., 2007) during localization. Additionally, I have demonstrated that particularized solutions create a large degree of interdependency between development nodes if introduced

globally, and how such dependencies slow down processes in GSW (Herbsleb and Mockus, 2003).

I have emphasized the need to verify localized information systems within their target context. In accordance with the research of Nhampossa (2004), I have confirmed the need for testing of linguistic terms with the target users of a system. Through an action research approach of the implementation of a HIS in Vietnam, pilot tests have been identified as an ideal framework in which to include such testing.

- *Local and global tensions in relation to software design.*

I have through the theories of Rolland and Monteiro (2002) and Pollock et al. (2007) investigated how particularized solutions can be avoided by striking a pragmatic balance between the sensitivity of local solutions and the need to standardize across contexts. Through alignment of interests and work towards generalized solutions, I have demonstrated that requirements emerging locally can be generalized into benefiting a project globally, and thus bridge local and global tensions. Based on findings in the context of the DHIS 2 application, cases where this has both failed and succeeded have been discussed.

Through the particularization of an installation procedure supporting local requirements, I have emphasized how the global perspective can be lost under time pressure locally and the extra labor involved in creating generalized solutions. I have in contrast to this case, identified several cases where local requirements have been negotiated and developed into generalized solutions that have diffused to benefit a project globally.

- *The establishment and integration of localization teams into global development networks.*

While my empirical study unveiled a digital divide technically, my research suggests that greater challenges emerge related to the effective use of technology in developing

countries. I have discussed the use of technology in the context of the two extremes of open source philosophy emerging from western culture and the strict hierarchical leadership structures in developing countries. By applying the term effective use as defined by Gurstein (2003), I have demonstrated how actors in developing countries were reluctant to effectively use communication forums with unfamiliar frame of reference (Edwards, 2001), that was introduced through an OSS project. My research strongly suggests that it is possible to a large extent to overcome this challenge by building personal relationships through participatory research processes such as action research. Based on my discoveries with the DHIS 2 application, I have identified how motivational factors in the adoption of OSS in developing countries (Weber, 2003) are compromised due to a lack of integration with development processes, and pointed to a generally low awareness of OSS in developing countries. Additionally, in light of an unsuccessful deployment of a centralized internet based solution, I have identified how effective localization solutions can be hindered by social structures.

I have also revealed how flexible standards enable localization processes to affect global requirements through technology choices, and how costs are a part of the development towards generalized and standardized solutions. My research suggests that it may be necessary for global software projects to strike balances between technical flexibility enabled by complex frameworks and a need to situate across all working conditions in a development network, both in regard to development frameworks and in regard to communication tools. Consequences if localization processes are particularized and independent processes "fork" have been emphasized. Based on a case from the development of DHIS 2, I have illustrated how such particularization can create tensions between developers globally and developers working with localization.

The focus on generality and abstractions proved applicable for my second research question as well, which in contrast to my primary research objective proved to be more of a technical challenge than a social challenge.

Secondary research objective: *Explore how internationalization can be enabled on the persistence layer of a software development project with an installed base.*

I have demonstrated using a generic aspect oriented concept that it is possible to enable the most widely used form of internationalization for persistent data, and aligning it with an existing installed base. A strive towards generality and abstractions were, in accordance with the conclusion of my primary research objective, also applicable for internationalization of the lower layers in software design. I have demonstrated how particularization (Pollock et al., 2007) was avoided, and confirmed assertion of Kersten et al. (2002) that aspect orientation is an effective approach to enable internationalization on the lower layers of applications.

7.1 Possible future research

In the DHIS 2 case technology that has been designed from the ground up to support internationalization aspects are used. Many systems do not have the option of redesign that DHIS had, and legacy systems may hence need implementation of localization. I suspect this to be very challenging tasks, especially because of the hacks performed in order to achieve some degree of localization and conversions of existing data. Having a name containing a Norwegian special character, I know that there are still a lot of systems out there that use legacy encoding schemes. Issues like the reserved ranges used to enable the variable width of Unicode present challenges in the conversion to and from other character systems.

Nhampossa (2004) emphasizes the spacing of the user interface as one issue when translating DHIS 1.4. This application is built around the same concepts (ref. 4.2.1) as my empirical base of DHIS 2, and we encountered no issues with spacing. Even when DHIS 2 was translated to a language using the non-Latin characters of the Hindi script language this did not pose any major challenges. This suggests that the web technology has an advantage over more statically sized desktop applications in internationalization development. This is however arguably a loose assumption, as more empirical evidence such as a comparison between the two products and platforms are needed to draw any conclusions.

I have also barely scratched the surface of aspects such as cultural adaptive software. It's widely established that there is a lot to gain on implementing such functionality, but still very few proposals on how to it can be achieved in a cost effective manner.

8. References

AltaVista (2007) Babel Fish Translation. Available: http://babelfish.altavista.com/ (Last accessed January 12, 2007)

ASF (2007) Maven introduction. Available: http://maven.apache.org/what-is-maven.html (Last accessed July 12, 2007)

Atlassian (2007) Confluence – the Enterprise Wiki. Available: http://www.atlassian.com/software/confluence/ (Last accessed July 12, 2007)

D. Avison, F. Lau, M. Myers & P.A. Nielsen (1999) Action Research. *Communication of the ACM,* January 1999/Vol. 42.

E. Baark & R. Heeks (1998) Evaluation of Donor-Funded Information Technology Transfer Projects in China: A Lifecycle Approach. *Development Informatics: Working Paper Series*

Baraha (2007) Baraha - Free Indian Language Software. Available: http://www.baraha.com/ (Last accessed January 20, 2007)

BBC-News (2003) Vietnam cuts priest's jail term. Available: http://news.bbc.co.uk/1/hi/world/asia-pacific/3074147.stm (Last accessed July 12, 2007)

BBC-News (2006) Vietnam gives Gates star welcome. Available: http://news.bbc.co.uk/2/hi/asia-pacific/4933290.stm (Last accessed June 25, 2007)

B. Bergstein (2007) Intel, `$100 Laptop' Project Make Peace. Available: http://news.wired.com/dynamic/stories/H/HUNDRED_DOLLAR_LAPTOP_INTEL?SITE=WIRE&SECTION=HOME&TEMPLATE=DEFAULT (Last accessed July 18, 2007)

J. Braa, O. Hanseth, A. Heywood, W. Mohammed & V. Shaw (2007) DEVELOPING HEALTH INFORMATION SYSTEMS IN DEVELOPING COUNTRIES: THE FLEXIBLE STANDARDS STRATEGY. *MIS Quarterly,* 31.

J. Braa & C. Hedberg (2002) The Struggle for District-based Health Information Systems in South Africa. *The Information Society*.

J. Braa, E. Monteiro & S. Sahay (2004) Networks of action: sustainable health information systems across developing countries. *MIS Quarterly*.

B. Chelf (2006) Insecurity in Open Source. Available: http://www.businessweek.com/technology/content/oct2006/tc20061006_394140.htm?campaign_id=bier_tco.g3a.rss1007 (Last accessed 2007, July 20)

A. Dix, J. Finlay, G.D. Abowd & R. Beale (2004) *Human-Computer Interaction*. isbn:0-13-046109-1

DriverHeaven (2007) Unsupported and bitter. Available: http://www.driverheaven.net/%7Epete/article5.htm (Last accessed July 20, 2007)

Edgewall.org (2007) The Trac project. Available: http://trac.edgewall.org/ (Last accessed July 13, 2007)

K. Edwards (2001) Epistemic Communities, Situated Learning and Open Source Software Development. *Epistemic Cultures and Practice of Interdisciplinarity, Workshop at NTNU (Trondheim).*

L. Elmer (2002) Vietnam's ICT Enabling Environment: Policy, Infrastructure and Applications. *U.S. Agency for International Development*.

B. Esselink (2000) *A practical guide to localization*, John Benjamins Publishing Company.

K. Fogel (2005) *Producing Open Source Software: How to Run a Successful Free Software Project*.

F.J. Fowler (2002) Survey Research Methods.

GALA (2007) The Globalization and Localization Association (GALA). Available: http://www.gala-global.org/who-we-are.html (Last accessed July 12, 2007)

J. Gallant & H. Bergevin (2007) How To Attack An Internet Explorer (Win) Display Bug. Available: http://www.communitymx.com/content/article.cfm?cid=C37E0 (Last accessed July 12, 2007)

M. Gurstein (2003) Effective use: A community informatics strategy beyond the Digital Divide. *First Monday,* 8.

HCMCPC (2007) Core facts. Available: http://www.eng.hochiminhcity.gov.vn/eng/news/ (Last accessed July 12, 2007)

J. Herbsleb & A. Mockus (2003) An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering,* 29.

K.G. Herr & G.L. Anderson (2005) *The Action Research Dissertation: A Guide for Students and Faculty*, Sage publications.

HISP (2007) The DHIS 2 Wiki. Available: http://www.hisp.info/confluence/display/DHIS2/Home (Last accessed July 12, 2007)

HISPIndia (2007) Graphical Analyzer. Available: http://hispindia.org/index.php?section=172 (Last accessed July 20, 2007)

Hydro (2007) Oil and gas activities to merge with Statoil. Available: http://www.hydro.com/northamerica/en/press_room/news_press_releases/2006_12/statoil_merger_en.html (Last accessed July 12, 2007)

I18nGurus (2007) Machine translations. Available: http://www.i18ngurus.com/docs/995054490.html (Last accessed July 12, 2007)

IBM (2007) Globalize your On Demand Business. Available: http://www-306.ibm.com/software/globalization/topics/translationgraphics/conclusion.jsp (Last accessed July 17, 2007)

ISO (2007a) ISO-3166: English country names and code elements. Available: http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html (Last accessed July 12, 2007)

ISO (2007b) ISO 3166-1 and country coded Top-Level Domains (ccTLDs). Available: http://www.iso.org/iso/en/prods-services/iso3166ma/04background-on-iso-3166/iso3166-1-and-ccTLDs.html (Last accessed July 12, 2007)

ISO (2007c) Overview of the ISO system. Available: http://www.iso.org/iso/en/aboutiso/introduction/index.html (Last accessed July 12, 2007)

ISO (2007d) Widely used standards: Language codes. Available: http://www.iso.org/iso/en/prods-services/popstds/languagecodes.html (Last accessed July 12, 2007)

R. Johnson (2005) Introduction to the Spring framework. Available: http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework (Last accessed July 12, 2007)

S. Joshi, M. Barett, G. Walsham & S. Cappleman (2006) In search of balance: local knowledge within global organisations. *Judge Business School, University of Cambridge*.

KDE (2007) KDE Localization. Available: http://l10n.kde.org/ (Last accessed July 12, 2007)

G.E. Kersten, M.A. Kersten & W.M. Rakowski (2002) Software and culture: Beyond the internationalization of the interface. *Journal of Global Information Management*.

R. Kling, H. Crawford, H. Rosenbaum, S. Sawyer & S. Weisband (2000) Learning from Social Infrastructures: Information and Communication Technologies in Human Contexts.

B. Kogut & A. Metiu (2001) OPEN-SOURCE SOFTWARE DEVELOPMENT AND DISTRIBUTED INNOVATION. *Oxford review of economic policy,* 17.

T. Kubota (2006) Debian: Introduction to i18n. Available: http://www.debian.org/doc/manuals/intro-i18n/ (Last accessed July 12, 2007)

M.L. Levin & D.J. Greenwood (1998) *Introduction to Action Research: Social Research for Social Change*, Sage Publications. isbn:0761916768

LibraryofCongress (2007) ISO 639.2: Frequently asked questions (FAQ). Available: http://www.loc.gov/standards/iso639-2/faq.html (Last accessed July 12, 2007)

Linuxlookup.com (2007) Compiz and Beryl reunited officially. Available: http://linuxlookup.com/2007/apr/05/compiz_and_beryl_reunited_officially (Last accessed July 20, 2007)

LISA (2007) The Localization Industry Standards Association. Available: http://www.lisa.org/info/faqs.html (Last accessed July 12, 2007)

A. MacCormack, J. Rusnak & C. Baldwin (2006) Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science,* 52.

M.J. Mahemoff & L.J. Johnston (1998) Software Internationalisation: Implications for Requirements Engineering *Deakin University: Geelong,* October 26-27**,** 83-90.

MarketShare (2007) Operating Systems Market Share For June 2007. Available: http://marketshare.hitslink.com/report.aspx?qprid=2 (Last accessed July 20, 2007)

N. Marsland, I. Wilson, S. Abeyasekera & U. Kleih (2001) COMBINING QUANTITATIVE (FORMAL) AND QUALITATIVE (INFORMAL) SURVEY METHODS *Natural Resources Institute*.

T. Mayeur (2007) Globalize your On Demand Business: Graphics Translation Process Overview. Available: (Last accessed July 17, 2007)

J.C. Mitchell (2003) *Concepts in programming languages*, Cambridge University Press. isbn:0-571-78098-5

D. Moore (2004) DNS server survey. Available: http://mydns.bboy.net/survey/ (Last accessed July 12, 2007)

Netcraft (2007) June 2007 Web Server Survey. Available: http://news.netcraft.com/archives/2007/06/08/june_2007_web_server_survey.html (Last accessed July 12, 2007)

J.L. Nhampossa (2004) THE CHALLENGE OF "TRANSLATING" HEALTH INFORMATION SYSTEMS FROM ONE DEVELOPING COUNTRY CONTEXT TO ANOTHER: CASE STUDY FROM MOZAMBIQUE.

J.L. Nhampossa & P. Nielsen (2004) Experiences of internationalizing Information Systems: The challenge of standardization. Cambridge.

P. Nielsen & J.L. Nhampossa (2005) Internationalization of Information Infrastructures and Control: Cases from Mozambique and Norway. *IFIP9.4.*

K. Nordal (2006) The Challenge of Being Open - Building an Open Source Development Network *Department of informatics, UiO.* Master thesis

J. O'Conner (2005) Internationalization: Understanding Locale in the Java Platform. Available: http://java.sun.com/developer/technicalArticles/J2SE/locale/index.html (Last accessed July 12, 2007)

R. O'Brien (1998) An Overview of the Methodical Approach of Action Research. *Faculty of Information Studies, University of Toronto*.

OLPC (2007a) OLPC Human Interface Guidelines. Available: http://wiki.laptop.org/go/OLPC_Human_Interface_Guidelines (Last accessed July 12, 2007)

OLPC (2007b) OLPCWiki: Not for individual sale. Available: http://wiki.laptop.org/go/Not_for_individual_sale (Last accessed July 13, 2007)

OSI (2007) Open Source: Frequently Asked Questions. Available: http://open1.mirrors-r-us.net/advocacy/faq.php (Last accessed July 12, 2007)

L.H. Øverland (2006) Global Software Development and Local Capacity Building: A means for improving Sustainability in Information Systems Implementations. *Department of informatics, UiO.* Master thesis

Pango.org (2007) Pango. Available: http://www.pango.org/ (Last accessed July 12, 2007)

N. Pollock, R. Williams & L. D'Adderio (2007) Global Software and its Provenance: Global Software and its Provenance. *Social Studies of Science,* 37.

E.S. Raymond (2000) *The Cathedral and the Bazaar*, O'Reilly. isbn:978-0596001315

K. Reinecke & A. Bernstein (2006) Culturally Adaptive Software: Moving Beyond Internationalization. Department of Informatics, University of Zurich.

G. Robles (2004) A Software Engineering approach to Libre Software -. *Jahrbuch*.

K. Rolland & E. Monteiro (2002) Balancing the local and the global in infrastructural information systems. *The Information Society*.

S. Sahay (2003) Global software alliances: the challenge of "standardization". *Scandinavian Journal of Information Systems,* 15**,** 18.

SCIM (2007) Smart Common Input Method platform. Available: http://www.scim-im.org/projects/imengines (Last accessed July 12, 2007)

SDS (2007) Installer2GO Home Page. Available: http://www.dev4pc.com/installer2go.html (Last accessed July 12, 2007)

L. Sherriff (2007) Negroponte slams Intel over OLPC competition. Available: http://www.theregister.co.uk/2007/05/21/olpc_vs_intel/ (Last accessed July 12, 2007)

Sun (2006) Locale. Available: http://java.sun.com/j2se/1.5.0/docs/api/java/util/Locale.html (Last accessed July 12, 2007)

Sun (2007a) Designing Enterprise Applications with the J2EETM Platform, Second Edition: EIS Tier Internationalization Available: http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/i18n/i18n5.html (Last accessed July 20, 2007)

Sun (2007b) The Java Tutorials: Formatting. Available: http://java.sun.com/docs/books/tutorial/i18n/format/index.html (Last accessed July 12, 2007)

B.A. Tate & J. Gehtland (2004) *Better, Faster, Lighter Java*, O'Reilly. isbn:0596006764

Tigris.org (2007) Subversion. Available: http://subversion.tigris.org/ (Last accessed July 12, 2007)

UnicodeConsortium (2007) What is Unicode? Available: http://www.unicode.org/standard/WhatIsUnicode.html (Last accessed July 12, 2007)

L. Vikør (2001) *The Nordic languages. Their Status and Interrelations*, Novus Press. isbn:82-7099-336-0

VNS (2007) Intel donates computers to promote tech-based learning. Available: http://vietnamnews.vnagency.com.vn/showarticle.php?num=01EDU090607 (Last accessed July 12, 2007)

W3 (2007a) Browser statistics. Available: http://www.w3schools.com/browsers/browsers_stats.asp (Last accessed July 12, 2007)

W3 (2007b) Cascading Style Sheets. Available: http://www.w3.org/Style/CSS/ (Last accessed July 17, 2007)

W3C (2007) W3C Sets New Standard for Internationalized Web Content. Available: http://www.w3.org/2007/04/its-pressrelease (Last accessed July 12, 2007)

S. Weber (2003) Open Source Software in Developing Economies. Berkley, University of California.

WebWork (2007) WebWork - WebWork. Available: http://www.opensymphony.com/webwork/ (Last accessed July 12, 2007)

WHO (2005) Review of health information systems (HIS) in selected countries. Available: (Last accessed July 12, 2007)

WHO (2007a) International Classification of Diseases (ICD). Available: http://www.who.int/classifications/icd/en/ (Last accessed June 26, 2007)

WHO (2007b) WHO and the Millennium Development Goals. Available: http://www.who.int/mdg/en/ (Last accessed July 12, 2007)

Wikibooks.org (2007) FOSS Localization/Localization Efforts in the Asia-Pacific. Available: http://en.wikibooks.org/wiki/FOSS_Localization/Localization_Efforts_in_the_Asia-Pacific (Last accessed July 12, 2007)

Wikipedia (2007a) Communism. Available: http://en.wikipedia.org/wiki/Comunist (Last accessed July 12, 2007)

Wikipedia (2007b) Create, read, update and delete. Available: http://en.wikipedia.org/wiki/Create%2C_read%2C_update_and_delete (Last accessed July 12, 2007)

Wikipedia (2007c) India. Available: http://en.wikipedia.org/wiki/India (Last accessed July 12, 2007)

Wikipedia (2007d) Internationalization and localization. Available: http://en.wikipedia.org/wiki/Internationalization (Last accessed July 12, 2007)

Wikipedia (2007e) Internet. Available: http://en.wikipedia.org/wiki/Internet (Last accessed July 12, 2007)

Wikipedia (2007f) Language localisation. Available: http://en.wikipedia.org/wiki/Language_localization (Last accessed July 12, 2007)

Wikipedia (2007g) Moore's Law. Available: http://en.wikipedia.org/wiki/Moores_law (Last accessed July 12, 2007)

Wikipedia (2007h) Vietnam. Available: http://en.wikipedia.org/wiki/Vietnam (Last accessed July 12, 2007)

Wikipedia.org (2007) The chicken or the egg. Available: http://en.wikipedia.org/wiki/The_chicken_or_the_egg (Last accessed 2007)

WTO (2006) ACCESSIONS: Viet Nam. Available: http://www.wto.org/english/thewto_e/acc_e/a1_vietnam_e.htm (Last accessed July 12, 2007)

XenCraft (2005) Whats wrong with Locales? Available: http://www.i18nguy.com/locales/Locales.pdf (Last accessed July 12, 2007)

A. Yeo (1996) Cultural User Interfaces - A Silver Lining in Cultural Diversity. *SIGCHI,* 28**,** 4.

9. Appendixes

Appendix A: Lists

Acronyms and abbreviations

AJAX	Asynchronous JavaScript and XML
AOP	Action Oriented Programming
ASF	Apache Software Foundation
AR	Action Research
BIND	Berkley Internet Name Domain
CCS	Coded Character Set
CES	Character Encoding Scheme
CHC	Commune Health Center
CSS	Cascading Style Sheets
DBMS	Database Management System
DHIS1	District Health Information Software version 1
DHIS1.x	District Health Information Software version 1 series
DHIS 2	District Health Information Software version 2
DNS	Domain Name Service
G11n	Globalization
GALA	The Globalization and Localization Association
GSW	Global Software Work
GUI	Graphical User Interface
HCi	Human-Computer interaction
HCMC	Ho Chi Minh City (formerly known as Saigon)
HIS	Health Information System
HISP	Health Information Systems Programme
HISPML	HISP Multilanguage Library
I18n	Internationalization
ICT	Information and Communication Technology

IM	Instant Messaging
IoC	Inversion of Control
IS	Information Systems
ISO	International Standards Organisation
JVM	Java Virtual Machine
L10n	Localization
LISA	Localization Industry Standards Association
MIT	Massachusetts Institute of Technology
MoH	Ministry of Health
MVC	Model View Controller
OLPC	One Laptop Per Child
ORM	Object/Relational Mapping
OS	Open Source
OSS	Open Source Software
POJO	Plain Old Java Object
POM	Project Object Model
UiO	University of Oslo
W3C	World Wide Web Consortium
WHO	World Health Organisation
WTO	World Trade Organisation

Figure list

Appendix B: Implementation plan

General

This plan is based on:

- Meetings with the Vietnamese team in HCMC during week 15-16, 2006.
- Discussions with Ola Hodne Titlestad and Knut Staring during week 15-16, 2006.
- Meetings with Dr. Chinh, Dr. Cong and Mr. Giang during week 16, 2006.

The plan

We have separated this initial deployment into two phases.

Phase 1: Preparation for installation

- Duration 2-3 weeks.
- Estimate: April 25th to May 12th (Independence day vacation during this period)
- Implement the system in Tan Binh District.
- Train users, one-by-one sessions.
- Build up a good relationship with the users. It's important that this first district has available time to give us proper feedback on the system.
- Get the system approved by both users and the Mother & Child Health Center, we do not move on to the next phase until we have this approval.

Phase 2: Pilot (expand)

- Duration: 2-3 months.
- Estimate: May 13th to the end of July.
- When we have "happy users" in the first district we expand.
- Implement in 2 more districts and 2 hospitals. Ideally at least one hospital should have a well functioning local network so we can test networking features.
- Do training and continue support (create manual, have courses, follow-up at the workplace)

Future progress

- After this we may expand further to step by step cover all districts.

- With this progress we hope to ensure that we can give proper support and training to all users.

Work routines

The team will work at an office at the Mother & Child Health Center and file individual reports weekly here. During these first phases the communication with the users is of particular importance. By being able to support the solution we can create a good relationship with the users. Previous experiences have shown that if we don't respond well to user requests they will stop reporting problems. It will be very important to identify problems as early as possible.

Appendix C: Survey

Please answer the questions by supplying a value after the question, feel free to leave comments after each question. I will use this in my master thesis, anonymity of all respondents is guaranteed in the sense that no names will be mentioned. Respondents may be referred to by role and country. Feel free to contact me on oyvind.brucker@gmail.com if you have any questions regarding this survey. All respondents may not have experience with all subjects here, so questions may be left blank.

Personalia

Please enter some personal information, feel free to enter multiple values for each question to best describe your role in the project.

Home country:
Country/Countries where you have worked with DHIS 2 locally:
Position(s):
Additional information (*optional*):

Part 1: User-interface translations

Please rate these statements on a scale of 1-6 (6 = true).

1.1 The DHIS 2 resource editor (the translator program) is easy to use: _____

1.2 The process of submitting translations to the project is straight forward: _____

1.3 I have not experienced input related and/or encoding related issues while translating (Text appeared correctly in DHIS 2 and/or the translation program after translation): _____

Part 2: DHIS 2 localization

Please rate these statements on a scale of 1-6 (6 = true).
By localization I refer to the ability of the software to adapt to the local context.

2.1 DHIS 2 is easy to set up and deploy locally: _____

2.2 The answers from the DHIS 2/HISP community are informative and helpful: _____

2.3 The response time from the DHIS 2 community is adequate: _____

2.4 DHIS 2 adapts well to local context(s) and users find it easy to use: _____

2.5 Local requests made to the DHIS 2 community are being heard: _____

2.6 DHIS 2 is working well on the computers where it is deployed (in terms of speed and other hardware requirements): _____

Part 3: Database Internationalization

Please rate these statements on a scale of 1-6 (6 = true).
This feature is not yet in use at all locations. This section refers to the ability to translate objects stored in the database, i.e. Organisation units and Data elements.

3.1 Its easy to translate objects in the database: _____

3.2 When upgrading existing installations to support database Internationalization no problems were encountered: _____

Part 4: Open

This part is open to any comments. Any issues big or small that have surfaced when deploying DHIS 2 locally, criticism is very much appreciated.